NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPMENT OF A HARDWARE-IN-THE-LOOP SIMULATOR FOR CONTROL MOMENT GYROSCOPE-BASED ATTITUDE CONTROL SYSTEMS

by

Brian C. Fields

December 2015

Thesis Advisor: Mark Karpenko
Second Reader: I. Michael Ross

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2015	**3. REPORT TYPE AND DATES COVERED** Master's thesis	
4. TITLE AND SUBTITLE DEVELOPMENT OF A HARDWARE-IN-THE-LOOP SIMULATOR FOR CONTROL MOMENT GYROSCOPE-BASED ATTITUDE CONTROL SYSTEMS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Brian C. Fields			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In this thesis, an open-architecture control moment gyroscope (CMG) system is developed for hardware-in-the-loop (HIL) simulation of spacecraft attitude control. This effort included construction of four single-gimbal CMGs, implementation of an attitude dynamics model, a quaternion error feedback control system, and a pseudoinverse CMG steering law on a real-time controller. The modular design of the embedded flight computer software allows for various parameters (such as the spacecraft inertia tensor, CMG rate limits, and control system gains) to be rapidly iterated and deployed for testing on physical hardware. Real-time communication with the CMG hardware is achieved via a Controller Area Network (CAN) bus; CMG commanding and telemetry sampling (including position, velocity, and current) can be performed at different sampling frequencies. The impact of sampling frequency on control law determinism and the CMG gimbal rest position (referred to as gimbal drift) is demonstrated. The HIL simulation testbed developed in this thesis allows future researchers to evaluate novel attitude control and CMG steering algorithms as well as optimal attitude guidance in a real-time, laboratory environment.

14. SUBJECT TERMS control moment gyroscopes, spacecraft attitude controls, hardware-in-the-loop simulation, real-time attitude control, quaternion feedback control, pseudoinverse steering, gimbal drift.			**15. NUMBER OF PAGES** 135
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT OF A HARDWARE-IN-THE-LOOP SIMULATOR FOR CONTROL MOMENT GYROSCOPE-BASED ATTITUDE CONTROL SYSTEMS

Brian C. Fields
Lieutenant, United States Navy
B.S., United States Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2015

Approved by: Mark Karpenko, Ph.D.
Thesis Advisor

I. Michael Ross, Ph.D.
Second Reader

Garth V. Hobson, Ph.D.
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, an open-architecture control moment gyroscope (CMG) system is developed for hardware-in-the-loop (HIL) simulation of spacecraft attitude control. This effort included construction of four single-gimbal CMGs, implementation of an attitude dynamics model, a quaternion error feedback control system, and a pseudoinverse CMG steering law on a real-time controller. The modular design of the embedded flight computer software allows for various parameters (such as the spacecraft inertia tensor, CMG rate limits, and control system gains) to be rapidly iterated and deployed for testing on physical hardware. Real-time communication with the CMG hardware is achieved via a Controller Area Network (CAN) bus; CMG commanding and telemetry sampling (including position, velocity, and current) can be performed at different sampling frequencies. The impact of sampling frequency on control law determinism and the CMG gimbal rest position (referred to as gimbal drift) is demonstrated. The HIL simulation testbed developed in this thesis allows future researchers to evaluate novel attitude control and CMG steering algorithms as well as optimal attitude guidance in a real-time, laboratory environment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACRONYMS

3DOF	three degree-of-freedom
ADCS	attitude determination and control system/subsystem
BLDC	brushless DC
CAD	computer-aided design
CAN	Controller Area Network
CMG	control moment gyroscope
cRIO	Compact Reconfigurable Input/Output
DC	direct current
DCM	direction cosine matrix
ECU	electronic control unit
EPOS	Easy Positioning System
FIFO	first in, first out
FPGA	field-programmable gate array
HIL	hardware-in-the-loop
ISO	International Organization of Standardization
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LVM	LabVIEW Measurement
MED	momentum exchange device
MATLAB	Matrix Laboratory
MDF	medium-density fiberboard
MILE	Maxon Inductive Little Encoder
NI	National Instruments
NPS	Naval Postgraduate School
PDO	Process Data Object
PLC	programmable logic controller
R-SAT	Reconfigurable Spacecraft Autonomy Testbed
RAM	random access memory
RCS	reaction control system
RPM	revolutions per minute

RTOS	real-time operating system
RW	reaction wheel
SI	International System of Units
STK	Systems Tool Kit
TAS	three-axis simulator
UI	user interface
USB	Universal Serial Bus
VI	Virtual Instrument
VDC	volts DC
VSCMG	variable-speed CMG

VARIABLES

β	CMG skew angle
δ, $\vec{\delta}$	CMG gimbal angle, CMG gimbal angle vector
$\dot{\delta}$, $\dot{\vec{\delta}}$	CMG gimbal rate, CMG gimbal rate vector
$\dot{\delta}_{max}$	CMG gimbal rate limit
Θ	momentum wheel angular position
θ	rotation angle
Ω_{RW}, Ω_{CMG}	momentum wheel angular velocity
$\vec{\omega}$	spacecraft angular velocity vector
A	CMG Jacobian matrix
$C_{B/N}$	rotation matrix from body frame (B) to inertial frame (N)
c	quaternion feedback controller gain
$\vec{d}$	distance vector
$\hat{e}$	Eigenaxis unit vector
$\vec{F}$	force vector
f	frequency
$\vec{g}$	gravitational acceleration vector
h , $\vec{h}$	angular momentum, angular momentum vector

h_{RW}, $\bar{h}_{RW}$	reaction wheel angular momentum, reaction wheel angular momentum vector
h_{CMG}, $\vec{h}_{CMG}$	angular momentum, angular momentum vector
I	3×3 identity matrix
I_{Gimbal}, I_{Wheel}	gimbal motor current, momentum wheel motor current
J_{RW}	momentum wheel rotational inertial
K_I	integral controller gain
K_P	proportional controller gain
k	quaternion feedback controller gain
M	singularity measure
m	mass
Q	quaternion matrix
q, $\vec{q}$	quaternion, quaternion vector
$\vec{q}_0$	initial quaternion vector
$\vec{q}_c$	commanded quaternion vector
$\vec{q}_e$	quaternion error vector
$\dot{q}$, $\dot{\vec{q}}$	quaternion rate, quaternion rate vector
$\vec{r}$	position vector (center of rotation to center of mass)
T, $\vec{T}$	torque, torque vector
t	time
u, $\vec{u}$	control torque, control torque vector

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

To my wife, Brigid: Thank you for your unwavering support. Despite late nights and long weekends, you encouraged me until the very end. You were always willing to listen and never let me take myself too seriously. Thank you for always supporting my passion for this project—I could not have done it without you.

To my thesis advisor, Dr. Mark Karpenko: Thank you for your mentorship. This year has been an incredible experience and would not have been possible without your guidance along the way. Thank you for never letting me believe the sky was falling.

To my second reader, Dr. I. Michael Ross: Thank you for your support throughout my time at NPS. You always find new ways to engage your students and stimulate our learning. Thank you for making my time here a most rewarding experience.

To Levi Owen in the NPS Machine Shop: Thank you for your patience as I learned my way through the machining process. Your work is simply incredible.

To Eugene Sio and Luis Flores at Maxon Motors, and Nicolas Webster, Sam Kaley, and the Applications Engineers at National Instruments: Thank you for your insight throughout the design process and patience during many hours of troubleshooting.

To my classmates: Thank you for your friendship, encouragement, and advice—I look forward to crossing paths with you in the years to come.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Spacecraft attitude control is an item of critical interest to the aerospace community. Nearly every modern spacecraft includes an attitude determination and control subsystem (ADCS) composed of attitude sensors, actuators, and a flight computer for signal processing and command generation [1]–[3]. The purpose of a spacecraft ADCS is to maintain a prescribed attitude in the presence of disturbance torques and to maneuver (slew) the spacecraft to a new desired attitude. Most early spacecraft utilized spin stabilization for attitude control [3]. In this approach, the spacecraft is spun about its major axis, resulting in a gyroscopic stiffness that resists external disturbances and maintains the spacecraft attitude [2], [3]. External torquers (such as thrusters) can be used to adjust the rate of rotation and the overall spacecraft attitude (as necessary) [3]. More recently, many spacecraft systems have adopted three-axis attitude control. Three-axis attitude control is particularly useful for space-borne remote sensing platforms based on the potential for increased pointing accuracy and dynamic maneuvering without the need to spin the entire spacecraft body (commonly referred to as the bus) [2], [3]. Since the entire bus is maneuvered (rather than gimbaling the payload sensor toward an object of interest), three-axis stabilized spacecraft can accommodate larger payloads with increased pointing precision [4].

Three-axis attitude control is accomplished by means of internal or external torque generating devices (torquers). External torquers control spacecraft angular velocity by changing the overall angular momentum of the spacecraft; internal torquers simply exchange angular momentum between the spacecraft body and the attitude control actuators [3]. External torquers include devices such as reaction control system (RCS) thrusters (which expel propellant to impart torque on a spacecraft) or magnetic torque rods (which generate torque as a function of the interaction between the magnetic field of the Earth and the current passing through a coil) [3]. The mission duration of spacecraft utilizing thrusters is often determined by the amount of onboard propellant. While magnetic torque rods rely on electrical power rather than propellant, their utility is limited by the spacecraft size, maneuverability and pointing requirements, as well as the altitude

of the spacecraft (which determines the strength of the magnetic field) [1], [3]. Internal torquers include momentum exchange devices such as reaction wheels or control moment gyroscopes (CMGs). These devices operate under the principle of conservation of angular momentum; if a spacecraft needs to rotate in one direction, an opposing control torque is generated to exchange momentum between the spacecraft and the momentum exchange device [1]–[4]. In a system with no external disturbance torques, the net angular momentum of the system can be expressed as follows [2]:

$$\vec{h}_{Body} + \vec{h}_{MED} = 0 \tag{1}$$

$$\dot{\vec{h}}_{Body} + \dot{\vec{h}}_{MED} = 0 \tag{2}$$

Equations (Eqns.) (1) and (2) serve as the foundation for spacecraft attitude dynamics and control law (discussed in detail in Chapter II).

A. MOMENTUM EXCHANGE DEVICES

Torque, T, is defined as the change in angular momentum, h, with respect to time. Reaction wheels impart torque on a spacecraft by increasing or decreasing the angular velocity of a momentum wheel (otherwise known as a rotor) [1]. The instantaneous angular momentum of a reaction wheel is the product of the rotational inertia, J_{RW}, and the angular velocity, Ω_{RW}, of the rotor [2].

$$h_{RW} = J_{RW}\Omega_{RW} \tag{3}$$

Torque is therefore proportional to the acceleration of the rotor, $\dot{\Omega}_{RW}$, and is expressed as $T_{RW} = J_{RW}\dot{\Omega}_{RW}$. Eqn. (3) describes the magnitude (scalar value) of angular momentum. The vector representation of angular momentum is simply the scalar value projected along the axis of rotation of the momentum wheel (see Figure 1). Since reaction wheels are fixed with respect to the spacecraft body axes, a minimum of three reaction wheels are required to generate an arbitrary three-dimensional torque (required for three-axis attitude control) [4]. Furthermore, since the loss of a single reaction wheel would result in

the loss of three-axis attitude control (potentially mission-ending), most spacecraft include a fourth reaction wheel for redundancy [3].

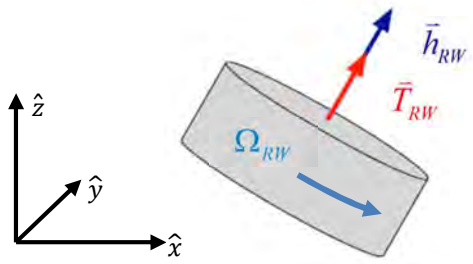

Figure 1. Reaction wheel torque and angular momentum vectors

Adapted from [5]: R. Votel and D. Sinclair, "Comparison of control moment gyros and reaction wheels for small Earth-observing satellites," in *26th Ann. AIAA/USU Conf. on Small Satellites*, Logan, UT, 2012.

A CMG is another type of momentum exchange device. Unlike reaction wheels, CMGs generate torque by gimbaling a momentum wheel rotating at a nominally fixed rate [2]. The torque output of a CMG is the cross product of the gimbal rate, $\dot{\vec{\delta}}$, and the instantaneous angular momentum of the momentum wheel, $\vec{h}_{CMG}$, resulting in a torque vector which is orthogonal to both the gimbal axis and the momentum wheel spin axis and proportional to the gimbal rate [5].

$$\vec{T}_{CMG} = \dot{\vec{\delta}} \times \vec{H}_{CMG} \tag{4}$$

The orientation of the output CMG torque vector with respect to the angular momentum vector and the gimbal axis (gimbal rate vector) is shown in Figure 2.

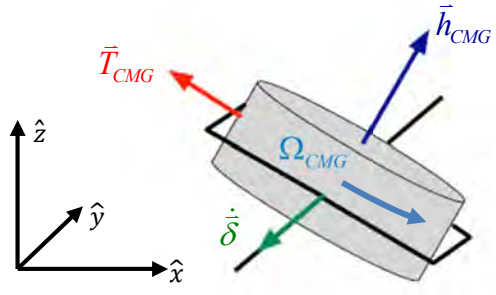

Figure 2. CMG torque and angular momentum vectors

Adapted from [5]: R. Votel and D. Sinclair, "Comparison of control moment gyros and reaction wheels for small Earth-observing satellites," in *26th Ann. AIAA/USU Conf. on Small Satellites*, Logan, UT, 2012.

Since the constant of proportionality for CMG torque is the angular momentum of the momentum wheel, CMGs are often referred to as torque amplifiers [4]. CMGs offer an increase in torque capability when compared with reaction wheels of the same size, mass, and power consumption (see Appendix B in [6] for a comparison of current reaction wheel and CMG characteristics). Likewise, CMGs offer a reduction in those same values when a specific torque capability is required, as compared to reaction wheels [7]. For these reasons, CMGs are a particularly appealing attitude control solution when designing agile spacecraft (such as commercial imaging satellites) and massive spacecraft (such as the International Space Station) [8]. In the case of imaging satellites, increased slew acceleration (a function of torque) can directly result in increased imaging opportunities and therefore increased revenue for the spacecraft operator [9], [10].

Despite being remarkably capable torquers, CMGs require a complex, non-intuitive control law which, if improperly implemented, can result in the spacecraft attitude control system encountering internal control singularities[1] well within the momentum envelope of the array [7]. Due to this complexity, the standard solution is to operate CMGs within a singularity-free region, which leverages only a small fraction of the available CMG capability (see [11] for example). To extend the range of operation of CMG systems, a variety of different singularity avoidance schemes have been proposed to augment the capability of CMG attitude control systems (see [12], [13], and [14] for examples of several different approaches). Part of the development of these new CMG steering strategies requires extensive computer simulation and ground-based testing of the resulting control algorithm in order to prove out the ideas. Subsequent hardware testing is necessary to fully vet control laws and ensure that physical CMGs behave as expected when the control algorithms are implemented on an embedded flight computer [15]. Ground testing also provides additional data on CMG power consumption and attitude control duty cycles for further engineering analysis.

[1] Chapter II provides a detailed discussion of CMG steering law and control singularities.

B. GROUND-BASED THREE-AXIS SIMULATORS

Ground-based attitude control testing is often performed using a three-axis simulator (see [16] for an extensive review of ground-based spacecraft simulators). Three-axis simulators include an array of attitude control actuators (typically reaction wheels or CMGs) and attitude sensors mounted on a three degree-of-freedom (3DOF) attitude stage. The attitude stage is generally afforded 30° to 45° of rotation about the pitch and roll axes and unlimited rotation about the yaw axis by means of a ball-and-socket air bearing [16]. Three-axis simulators provide the greatest analog to the attitude dynamics in the frictionless and micro-gravity environment of space, but also require precise balancing of the attitude stage and alignment of attitude control sensors and actuators [16], [17]. Any offset between the center of rotation and the center of mass results in an undesirable gravity torque. This artifact of ground-based testing results in a spurious accumulation of momentum and can result in the attitude control system operating outside of the expected or desired range [17], [18], thus circumventing the utility of a ground-based test program.

To illustrate the deleterious effect of an unbalanced system, consider the fact that torque, $\vec{T}$, is the result of a force, $\vec{F}$, applied some distance, $\vec{d}$, from a point of rotation. The components of torque (expressed in three dimensions) are determined by the vector cross-product:

$$\vec{T} = \vec{F} \times \vec{d} \tag{5}$$

For a three-axis simulator, the unbalance force in Eqn. (5) is the product of gravitational acceleration and the mass of the attitude stage. The lever arm is the position vector ($\vec{r}_{Offset}$ in Eqn. (6)) which describes the location of the center of mass with respect to the center of rotation[2] [18].

[2] The torque vector calculated using Eqn. (6) is expressed in the simulator body frame. This approach requires that the gravitational acceleration vector be expressed in the simulator body frame as well. Chapter II provides a more detailed discussion of reference frames, attitude parametrization, and transformation between inertial and body frames.

$$\begin{aligned}\vec{T} &= m\vec{a} \times \vec{r}_{Offset} \\ &= m \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}\end{aligned} \tag{6}$$

The disturbance torque expressed by Eqn. (6) results in an undesirable change in the angular momentum of system. To prevent this accumulation of spurious angular momentum, extensive efforts have been undertaken to automate the process of mass balancing (see [17] and [18] for two examples), however these automatic mass balancing systems add to overall complexity of three-axis simulators and require the development of additional control law and hardware interfaces. Whether accomplished manually or automatically, mass balancing remains a critical and time consuming component of ground testing using three-axis simulators, which can make the development of other techniques for proving out the performance of a spacecraft ADCS desirable.

C. HARDWARE-IN-THE-LOOP (HIL) SIMULATION

As a compromise to a full three-axis simulation, a hardware-in-the-loop (HIL) simulation can be utilized for control law prototyping and initial ground-based testing of attitude control maneuvers. In an HIL setup, a flight computer is used to provide real-time control inputs to physical CMGs (hardware) while a numerical software model simulates the dynamics of the spacecraft in response to the real CMG outputs [15]. The principal benefit of HIL testing is the ability to exercise real CMG hardware (with real dynamics and off-nominal nuances) without the need to fully develop a three-axis simulator and its necessary ancillary functions (such as mass balancing and momentum management). Furthermore, when using a three-axis simulator to evaluate an attitude control maneuver, the simulation assumes the mass properties of the attitude stage. Since the spacecraft attitude dynamics are modeled during an HIL test, the mass and inertia properties of the simulation can be adjusted to represent any spacecraft. Finally, physical testing (especially with a three-axis simulator) introduces a variety of sources of error (such as error and uncertainty in the mass and inertia properties of the simulator and the alignment of the CMGs with respect to the simulator body axes). While these errors are

important considerations in the ultimate implementation of the attitude control system, they can unnecessarily complicate the analysis of maneuvers during the early stages of development and testing.

D. THESIS OBJECTIVES AND SCOPE

In 2009, the Control and Optimization Laboratories at the Naval Postgraduate School (NPS) acquired a 3DOF Satellite Simulator from Andrews Space. This three-axis simulator, known as the NPS Reconfigurable Spacecraft Autonomy Testbed (R-SAT), is composed of an air-bearing attitude stage and base pedestal, as depicted in Figure 3. Figure 3 also depicts the orientation of the simulator body axes (of note, the $+\hat{Z}$ axis is pointing towards the ground). R-SAT attitude control was performed by four variable-speed CMGs (VSCMGs) and eight pneumatic RCS thrusters. Attitude determination was accomplished using three KVH DSP-3000 fiber optic gyros, a sun sensor, and a magnetometer. Unfortunately, two of the VSCMG momentum wheels experienced internal component failures, making three-axis attitude control impossible and crippling the experimental test facility.

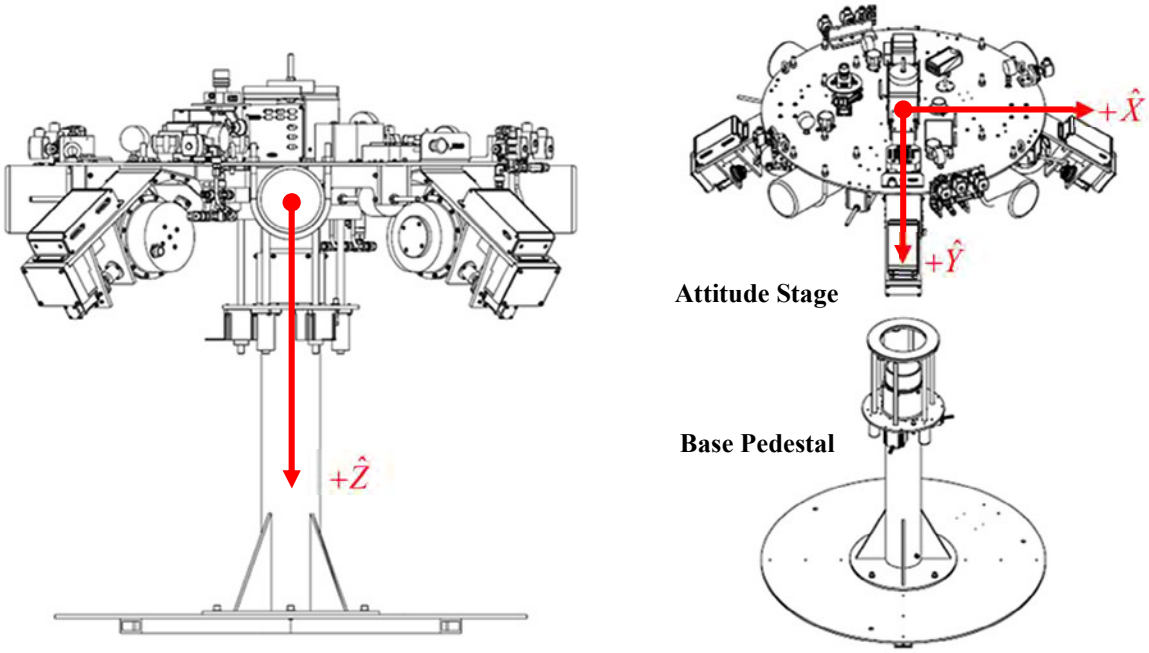

Figure 3. Overview of Andrews Space 3DOF Satellite Simulator

Adapted from [19]: *3DOF Satellite Simulator User's Guide*, Andrews Space, Tukwila, WA, 2009.

In 2012, an effort was undertaken to design and manufacture a drop-in momentum wheel enclosure to replace the failed units [20]. This research resulted in the production and characterization testing of a prototype momentum wheel. In 2014, a reconfigurable gimbal assembly was designed [21] which, together with the new momentum wheel design, could function as a complete replacement CMG. The new gimbal design addressed several shortcomings of the Andrews Space VSCMG design, most notably the fixed skew angle of the original system. The goal of this research is to build upon the previous redesign efforts and develop a four-CMG HIL simulation testbed for rapid prototyping of CMG control laws and attitude guidance schemes. This testbed includes an embedded flight computer running custom software to simulate spacecraft dynamics and perform real-time attitude control and CMG steering. All components of the HIL testbed are designed for eventual integration with the NPS R-SAT simulator. The inclusion of an attitude dynamics model on the embedded flight computer will allow the revitalized system to be used for both three-axis and HIL simulation testing. Data collected from a series of HIL experiments using the developed testbed are presented to illustrate the complexity and nuance of real-world systems and help underscore the criticality of ground-based testing before implementing control law on operational spacecraft.

E. THESIS OUTLINE

Chapter II provides a brief overview of spacecraft attitude dynamics, the quaternion feedback control law, and standard pseudoinverse CMG steering. Chapter III describes the physical hardware and communications architecture of the HIL testbed. Chapter IV discusses the implementation of the simulated spacecraft dynamics, attitude control law, and CMG steering law on a real-time controller as well as the logic for commanding the physical CMGs, collecting simulation telemetry, and displaying the simulated spacecraft attitude for motion visualization. Chapter V presents the results from a series of HIL experiments and attitude control maneuver simulations and discusses differences between software-only and HIL simulation results to illustrate the need for testing in a hardware environment. Finally, Chapter VI provides a summary of the work performed and recommendations for future work.

II. ATTITUDE DYNAMICS AND CONTROL

In order to control the orientation and motion of a spacecraft (or three-axis simulator), we must first describe the orientation of the spacecraft with respect to some reference frame. This chapter discusses the various approaches to attitude parametrization and develops the equations that govern spacecraft motion. Next, an approach to spacecraft attitude control known as quaternion feedback is discussed. A simple CMG steering law, required to map spacecraft attitude control commands to the necessary motor commands for CMG gimbal actuation, is also presented. The equations developed in this chapter form the basis for the implementation of the spacecraft dynamics and control model implemented later in the HIL system.

A. ATTITUDE DYNAMICS

Attitude describes the orientation of a rigid body with respect to a reference frame [1]. The reference frame is typically an inertial system defined by three orthogonal axes. There are multiple approaches to describing spacecraft attitude, each with inherent advantages and disadvantages. Among the most common attitude representations are direction cosines, sequential orthogonal rotations, and Euler parameters (more commonly referred to as quaternions) [2].

If it is assumed that the inertial reference frame (denoted by the subscript N) consists of three orthogonal axes, $\begin{bmatrix} \hat{i} & \hat{j} & \hat{k} \end{bmatrix}$, then the orientation of a spacecraft body (denoted by the subscript B) can be described by the cosines of the angles between the spacecraft body axes, $\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \end{bmatrix}$, and the basis vectors of the reference frame [4]. This relationship can be expressed as:

$$C_{B/N} = \begin{bmatrix} \hat{x}\cdot\hat{i} & \hat{x}\cdot\hat{j} & \hat{x}\cdot\hat{k} \\ \hat{y}\cdot\hat{i} & \hat{y}\cdot\hat{j} & \hat{y}\cdot\hat{k} \\ \hat{z}\cdot\hat{i} & \hat{z}\cdot\hat{j} & \hat{z}\cdot\hat{k} \end{bmatrix} \tag{7}$$

The rotation matrix in Eqn. (7) is defined as a direction cosine matrix (DCM), where the subscript B/N denotes the transformation from the body frame to the inertial frame. Thus, premultiplying a vector expressed in the body frame by the DCM, $C_{B/N}$, transforms the vector to the inertial reference frame. Likewise, the matrix transpose of the DCM, ${C_{B/N}}^T$, can be used to transform a vector from the inertial reference frame to the spacecraft body frame [4]. This simple duality is the result of the special properties of rotation matrices, namely $C^T C = I$, where I is the 3×3 identity matrix [22].

Whereas DCMs can be used to describe any arbitrary attitude, an Euler angle describes the angle of rotation of a body about a single axis and is typically denoted by the Greek letter theta, θ [1]. A rotation is referred to as an elementary rotation when it is about a single principal axis in the reference frame. When successive rotations are implemented, a numerical subscript can be used to denote the axis with which the individual Euler angles are associated (such as θ_1 describing a rotation about the first axis). The DCMs for elementary rotations are as follows [4]:

$$C_1(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \tag{8}$$

$$C_2(\theta_2) = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & \sin\theta_1 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \tag{9}$$

$$C_3(\theta_3) = \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Any non-elementary attitude can be achieved by performing multiple elementary rotations in series. The resulting DCM for a series of sequential rotations is achieved by premultiplying the respective DCMs defined in Eqns. (8), (9), and (10) in the order the rotations are performed [2].

A specific sequential rotation known as a 313 Euler rotation sequence involves elementary rotations about the third, first, and third axes. The Euler angles associated with the 313 Euler rotation sequence may also be denoted by the Greek letters $\theta_1 = \phi$, $\theta_2 = \theta$, and $\theta_3 = \psi$ [2]. The resulting DCM for a 313 Euler rotation sequence is as follows [4]:

$$C_{313} = C_3(\psi) C_1(\theta) C_3(\phi)$$

$$= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & \sin\theta\sin\psi \\ -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & \sin\theta\cos\psi \\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{bmatrix}$$

A total of 12 different rotation sequences exist (including the 123 rotation sequence, more commonly known as the roll, pitch, and yaw sequence). One disadvantage of Euler rotations is the existence of trigonometric singularities. The angles causing these singularities vary depending on rotation sequence. For example, the 313 Euler rotation sequence encounters a singularity when $\theta = 0$ or π, whereas the 123 rotation sequence encounters a singularity when $\theta_2 = \pi/2$ or $3\pi/2$ [22].

Expanding upon the concept of Euler angles, Euler's rotation theorem states a body can be reoriented (or its attitude parameterized) by rotating about a single axis which is fixed in both reference frames [2], [8] (such as the inertial and spacecraft body frames). This specific type of maneuver is known as an Eigenaxis rotation; the axis of rotation is referred to as the Eigenaxis (expressed as $\hat{e} = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}$) and the angle of rotation is referred to as the unit Eigenangle (once again represented by the Greek letter θ) [4]. Figure 4 provides an visualization of an Eigenaxis rotation with respect to two reference frames.

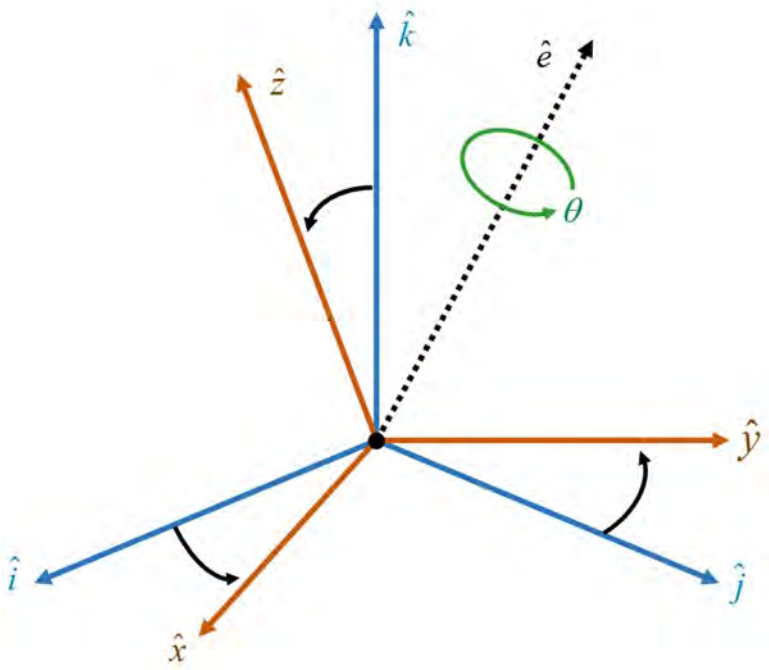

Figure 4. Visualization of example Eigenaxis rotation

Adapted from [8]: N. S. Bedrossian, S. Bhatt, W. Kang, and I. M. Ross, "Zero-propellant maneuver guidance: rotating the International Space Station with computational dynamic optimization," *IEEE Control Systems Magazine*, October 2009.

Euler parameters (more commonly referred to as quaternions) can be implemented based on the concept of Eigenaxis rotations. A quaternion ($\vec{q}$) is defined as follows:

$$\vec{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} e_1 \sin(\theta/2) \\ e_2 \sin(\theta/2) \\ e_3 \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix} \tag{12}$$

Additionally, quaternions are bound by the constraint $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$ [4]. Quaternions offer several advantages over the Euler rotation sequences described previously, most notably their lack of a trigonometric singularity and the reduction in computational complexity achieved by minimizing the use of trigonometric functions [1], [22]. This reduction in complexity is advantageous in real-time control laws. Since the goal of this thesis is to implement and execute a spacecraft control law on a real-time

controller, quaternions are used to represent the spacecraft attitude. This choice is also consistent with the state of practice in the aerospace industry.

Attitude kinematics deal with the change in attitude with respect to time (without concern for the cause of rotation). The time derivative of the quaternion ($\dot{\vec{q}}$) can be calculated based on the current quaternion (q) and the angular velocity of the body with respect to the inertial frame (referred to as $\vec{\omega}_{BN} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$) [2].

$$\dot{\vec{q}} = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{13}$$

Using Eqn. (13), the attitude of the spacecraft body can be determined by integrating the spacecraft quaternion rates, which are in turn obtained by integrating the spacecraft dynamic equations of motion. From the point of view of spacecraft attitude control, however body rates are typically measured using rate gyros so that Eqn. (13) can be integrated directly.

The rate of rotation of a body is a function of the angular momentum and the inertia of that body [2]. In order to change the rate of rotation of a body, a torque (change in angular momentum) must act upon the system. This torque can come in the form of an undesirable disturbance torque and/or applied control torques acting on the system.

The angular momentum of a body (about three axes) is a function of the angular velocity and the inertia of the body. This relationship can be described as follows:

$$\vec{h}_{Body} = J\omega_{B/N} \tag{14}$$

where J is the spacecraft inertia tensor.

Torque is defined as the change in angular momentum with respect to time. This relationship is expressed mathematically as [2]:

$$\vec{T}_N = \frac{d\vec{H}_N}{dt} \tag{15}$$

Eqn. (15) is expressed with respect to the inertial reference frame, but torque can be expressed in the spacecraft body frame (by way of the transport theorem) as [4]:

$$\vec{T}_B = J\dot{\vec{\omega}}_{B/N} + \vec{\omega}_{B/N} \times \left(J\vec{\omega}_{B/N} + \vec{h}_{CMG} \right) \tag{16}$$

Assuming perfect exchange of momentum (such that $\vec{T}_B = -\vec{T}_{CMG}$) and zero external disturbance torques, the angular acceleration of the spacecraft body can be determined by manipulating Eqn. (16) as follows:

$$\dot{\vec{\omega}}_{B/N} = J^{-1}\left(-\vec{T}_{CMG} - \vec{\omega}_{B/N} \times \left(J\vec{\omega}_{B/N} + \vec{h}_{CMG} \right) \right) \tag{17}$$

B. ATTITUDE CONTROL

There are a variety of different approaches for formulating a spacecraft control law, however a common approach is to use a quaternion feedback law[3]. A quaternion feedback control law generates torque commands as a function of the error, $\vec{q}_e$, between the current quaternion, $\vec{q}(t)$, and the commanded quaternion, $\vec{q}_c$ [4]. The quaternion error is expressed as follows [23]:

$$\vec{q}_e = \begin{bmatrix} q_{1e} \\ q_{2e} \\ q_{3e} \\ q_{4e} \end{bmatrix} = \begin{bmatrix} q_{4c} & q_{3c} & -q_{2c} & -q_{1c} \\ -q_{3c} & q_{4c} & q_{1c} & -q_{4c} \\ q_{2c} & -q_{1c} & q_{4c} & -q_{3c} \\ q_{1c} & q_{2c} & q_{3c} & q_{2c} \end{bmatrix} \begin{bmatrix} q_1(t) \\ q_2(t) \\ q_3(t) \\ q_4(t) \end{bmatrix} \tag{18}$$

The required control torque, $\vec{u}$, is then determined as [23]:

$$\vec{u} = -kJ\vec{q}_e - cJ\vec{\omega} + \vec{\omega} \times J\vec{\omega} \tag{19}$$

The constants k and c in Eqn. (19) are positive scalars, which dictate the characteristics (rise time and overshoot) of the second-order attitude response. As is seen, these gains are scaled by the spacecraft inertia tensor so that the response for each axis is

[3] The modular nature of the flight computer control software (as discussed in Chapter IV) allows for alternate control laws to be implemented with minimal impact to the existing design.

similar [23]. It is also possible to augment Eqn. (19) to include commanded rate terms, $\vec{\omega}_c$, and an acceleration feed forward command, $\vec{\alpha}_c$, as:

$$\vec{u} = -kJ\vec{q}_e - cJ\left(\vec{\omega} - \vec{\omega}_c\right) + J\vec{\alpha}_c + \vec{\omega} \times J\vec{\omega} \tag{20}$$

This latter form of quaternion feedback control is of particular use for executing a specified maneuver trajectory.

C. CMG STEERING

The control torques produced by Eqn. (19) or (20) are given with respect to the spacecraft body axes. The role of CMG steering law is to transform the desired 3×1 control torque vector (given in the body frame) to an $n\times1$ gimbal rate command vector for the individual CMGs (where n is the number of CMGs in the array). For the purposes of this thesis, a standard pseudoinverse steering law was implemented. More elaborate schemes are also possible (see [12], [13], and [14] for several examples) and, as will be seen later, the embedded flight computer control system architecture developed for the HIL testbed allows for easy implementation of any these algorithms.

To construct the standard CMG steering law, it is first necessary to determine the net CMG momentum in the body frame. The net CMG momentum is a function of the CMG skew angle, β, the rotational inertia, J_{CMG}, and angular velocity, Ω_{CMG}, of each momentum wheel, and the current gimbal angle of each CMG, δ_{CMG}. For a standard pyramidal configuration (see Figure 5), the net CMG angular momentum is given as follows [4]:

$$\begin{aligned}\bar{h} = J_1\Omega_1\begin{bmatrix} -\cos\beta\sin\delta_1 \\ \cos\delta_1 \\ \sin\beta\sin\delta_1 \end{bmatrix} + J_2\Omega_2\begin{bmatrix} -\cos\delta_2 \\ -\cos\beta\sin\delta_2 \\ \sin\beta\sin\delta_2 \end{bmatrix} + \ldots \\ J_3\Omega_3\begin{bmatrix} \cos\beta\sin\delta_3 \\ -\cos\delta_3 \\ \sin\beta\sin\delta_3 \end{bmatrix} + J_4\Omega_4\begin{bmatrix} \cos\delta_4 \\ \cos\beta\sin\delta_4 \\ \sin\beta\sin\delta_4 \end{bmatrix}\end{aligned} \tag{21}$$

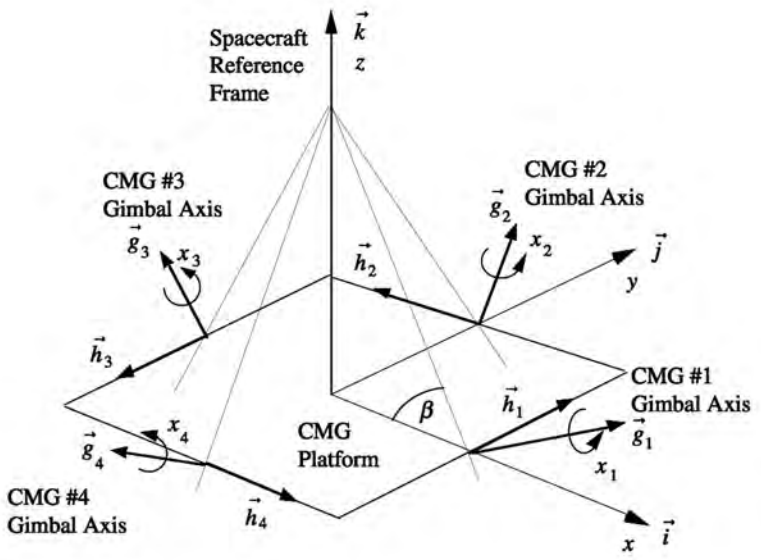

Figure 5. Pyramidal arrangement of four single-gimbal CMGs

Source [4]: B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed. Reston, VA: American Institute of Aeronautics and Astronautics, Inc., 2008, p. 670.

Having described the angular momentum in the body frame, it is now possible to construct the steering law. The control torque, $\vec{u}$, calculated using the quaternion feedback control law given in Eqn. (19) or (20) can be viewed as the time derivative of the CMG angular momentum vector, $\dot{\vec{h}}$. This derivative, by way of the chain rule, can be expressed as follows [4]:

$$\dot{\vec{h}} = \frac{d\vec{h}}{dt} = \frac{\partial \vec{h}}{\partial \delta}\frac{\partial \delta}{\partial t} \tag{22}$$

The partial derivatives of the angular momentum vector, $\vec{h}$, with respect to the CMG gimbal angles, δ, can be assembled into the so-called Jacobian matrix [4].

$$A = \begin{bmatrix} \frac{\partial \vec{h}_1}{\partial \delta_1} & \frac{\partial \vec{h}_1}{\partial \delta_2} & \frac{\partial \vec{h}_1}{\partial \delta_3} & \frac{\partial \vec{h}_1}{\partial \delta_4} \\ \frac{\partial \vec{h}_2}{\partial \delta_1} & \frac{\partial \vec{h}_2}{\partial \delta_2} & \frac{\partial \vec{h}_2}{\partial \delta_3} & \frac{\partial \vec{h}_2}{\partial \delta_4} \\ \frac{\partial \vec{h}_3}{\partial \delta_1} & \frac{\partial \vec{h}_3}{\partial \delta_2} & \frac{\partial \vec{h}_3}{\partial \delta_3} & \frac{\partial \vec{h}_3}{\partial \delta_4} \end{bmatrix} \tag{23}$$

Differentiating the net CMG angular momentum expressed in Eqn. (21) by the CMG gimbal angles gives the Jacobian matrix as follows:

$$A = \begin{bmatrix} -\cos\beta\cos\delta_1 & \sin\delta_2 & \cos\beta\cos\delta_3 & -\sin\delta_4 \\ -\sin\delta_1 & -\cos\beta\cos\delta_2 & \sin\delta_3 & \cos\beta\cos\delta_4 \\ \sin\beta\cos\delta_1 & \sin\beta\cos\delta_2 & \sin\beta\cos\delta_3 & \sin\beta\cos\delta_4 \end{bmatrix} \times \ldots$$
$$\begin{bmatrix} J_1\Omega_1 & 0 & 0 & 0 \\ 0 & J_2\Omega_2 & 0 & 0 \\ 0 & 0 & J_3\Omega_3 & 0 \\ 0 & 0 & 0 & J_4\Omega_4 \end{bmatrix} \tag{24}$$

The second partial derivatives expressed in Eqn. (22), $\partial\delta/\partial t$, are simply the individual gimbal rates, $\dot{\delta}$. Thus, using Eqn. (21), we can express the rate of change of angular momentum as $\dot{\bar{h}} = A\dot{\bar{\delta}}$. The gimbal rates are the control variables for the individual CMGs and their values can be determined using the Moore-Penrose pseudoinverse:

$$\dot{\bar{\delta}} = A^{+}\dot{\bar{h}} \tag{25}$$

where the Moore-Penrose pseudoinverse (A^{+}) is defined as [24]:

$$A^{+} = A^{T}\left(AA^{T}\right)^{-1} \tag{26}$$

The Moore-Penrose pseudoinverse represents the least squares solution to the control allocation problem. It should be apparent to the reader that the inverse $\left(AA^{T}\right)^{-1}$ in Eqn. (26) must exist in order to map the commanded torque to the individual gimbal rates. However, since A (and A^{T}) are functions of δ, the matrix A may become singular, leading to $\dot{\delta} = \infty$, which is not a practical solution. This potential breakdown in a standard CMG steering law results in the perceived need for more elaborate control strategies. As previously noted, only the standard steering scheme is used for this thesis, necessitating the operation of the CMG system within a restricted envelope [11].

D. CMG SINGULARITIES AND SINGULARITY AVOIDANCE

Two types of CMG control singularities exist. The extreme case exists at the boundary of the CMG momentum envelope (in which case the requested momentum exceeds the performance capabilities of the CMG array). The second case, known as an internal control singularity, occurs when the torque vectors of all CMGs are perpendicular to the commanded torque [4]. In the event of a control singularity, no finite control solution exists (due to AA^T in Eqn. (26) being non-invertible). Figures 6a and 6b show the normalized singularity surface for a pyramidal array of four single-gimbal CMGs with skew angles of 54° and 90°, respectively. In order to avoid singularities, it is necessary to operate within the singularity-free region (within the structure of the internal singularity surfaces). As can be seen, however, a significant amount of capability is lost when using this approach. In many practical systems, simplicity of the control system is traded for CMG capability (and therefore spacecraft agility). The plots in Figure 6 can also be used to help visualize the trajectory of an attitude control maneuver in momentum space (to evaluate the potential for encountering singularities). Such visualizations are presented in Chapter V using the results obtained from HIL experiments.

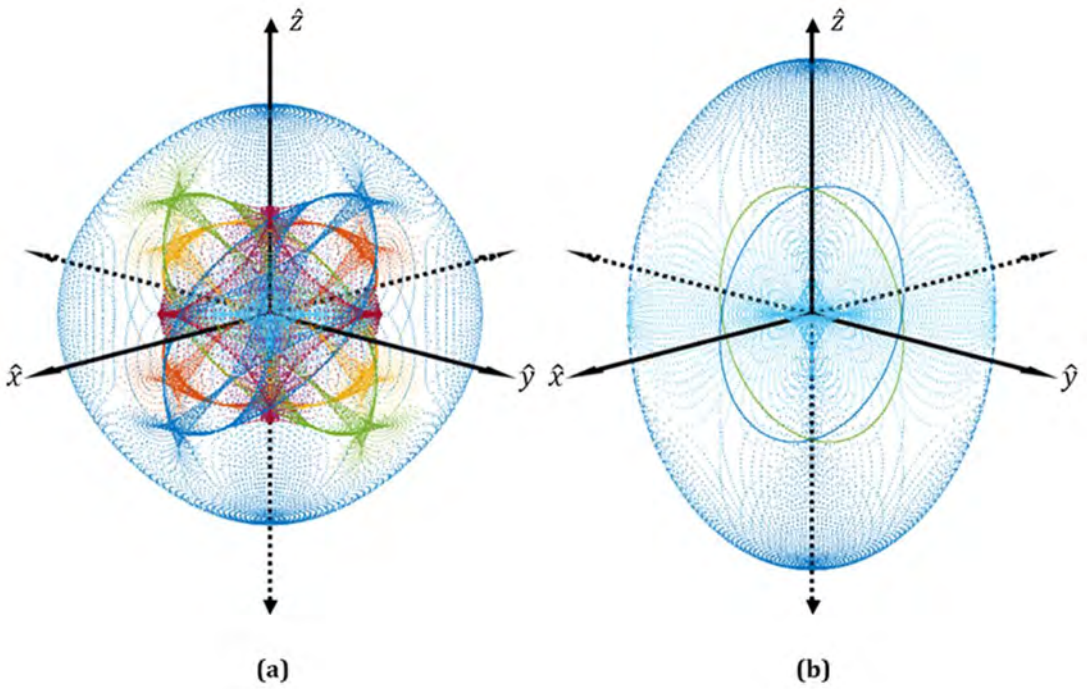

Figure 6. Normalized singularity surface for a pyramidal array of four single-gimbal CMGs

Adapted from [17]: J. Kim and B. Agrawal, "System identification and automatic mass balancing of a ground-based three-axis spacecraft simulator," in *AIAA Guidance Navigation, and Control Conference*, Keystone, CO, 2006.

The singularity measure, M, quantifies the degree to which the CMG array is approaching a singularity [4] and can be used as a real-time metric to evaluate whether the control solution is trending toward or away from a singular state. The singularity measure is calculated as follows:

$$M = \det\left(AA^T\right) \tag{27}$$

It is apparent from Eqn. (27) that $M = 0$ implies that the Moore-Penrose pseudoinverse cannot be computed and no control solution exists. Therefore, for predictable operation of the attitude control system, $M >> 0$ is desired. The singularity measure is thus one telemetry point that will be recorded as part of the HIL simulation experiments.

E. SUMMARY

Several approaches for attitude parameterization as well as the attitude kinematics and the equations of motion for attitude dynamics were presented in this chapter. Quaternion feedback was described and will be the solution for spacecraft attitude control implemented in the HIL testbed. A standard pseudoinverse CMG steering law will also be implemented despite the possibility of encountering singular states. The equations presented in this chapter will be revisited in Chapter IV when the logic for the attitude dynamics model, quaternion feedback control law, and CMG steering law will be implemented on a real-time controller using *LabVIEW*[4] Systems Design Software.

[4] For the remainder of this thesis, italics will be used to denote software titles, functions, and applications. Quotation marks will be used to denote user inputs and custom-generated functions within the embedded flight computer control software.

III. ARCHITECTURE OF THE HIL TESTBED

Previous student-led research at NPS resulted in the design and manufacture of an open-architecture, single-gimbal CMG prototype [20], [21]. This thesis expands on previous work by replicating a slightly modified version of the CMG prototype design and developing an HIL control architecture whereby a four-CMG array can be commanded to perform various spacecraft attitude maneuvers in real time. The HIL testbed includes a host workstation (for developing, deploying, and controlling simulation files), a real-time controller (functioning as an embedded flight computer), a DC power supply and power distribution bus, four open-architecture CMGs, and a communications bus for commanding the system in real time. The ultimate aim of this thesis is to replace the existing Andrews Space VSCMGs (and associated flight computer) on the NPS R-SAT simulator (see Figure 7). As discussed in Chapter I, the CMGs used in this thesis address several shortcomings of the Andrews Space VSCMG design, notably the fixed skew angle (β in Figure 7). However, the HIL testbed developed in this work is important in its own right. This chapter details the hardware configuration and communications architecture of the HIL testbed; design of the embedded flight computer control software will be discussed separately in Chapter IV.

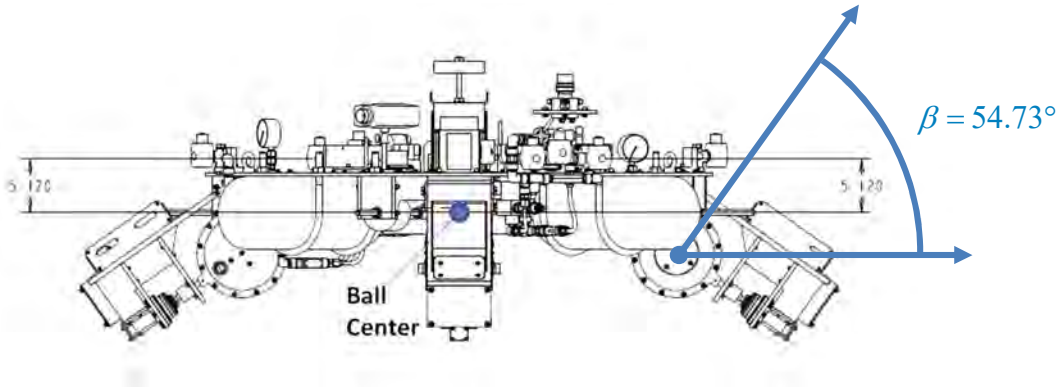

Figure 7. Side view of NPS R-SAT simulator

Adapted from [19]: *3DOF Satellite Simulator User's Guide*, Andrews Space, Tukwila, WA, 2009.

A. TESTBED LAYOUT AND SIGNAL FLOW DIAGRAM

Figure 8 shows a photograph of the CMG HIL testbed. All components of the HIL testbed, including the real-time controller, power distribution bus, and four single-gimbal CMGs, are mounted on a medium-density fiberboard (MDF) platform supported by an aluminum extrusion frame. Figure 9 provides a power and communications signal flow diagram. A description of each component, along with an overview of the communications architecture, is provided in the section that follows.

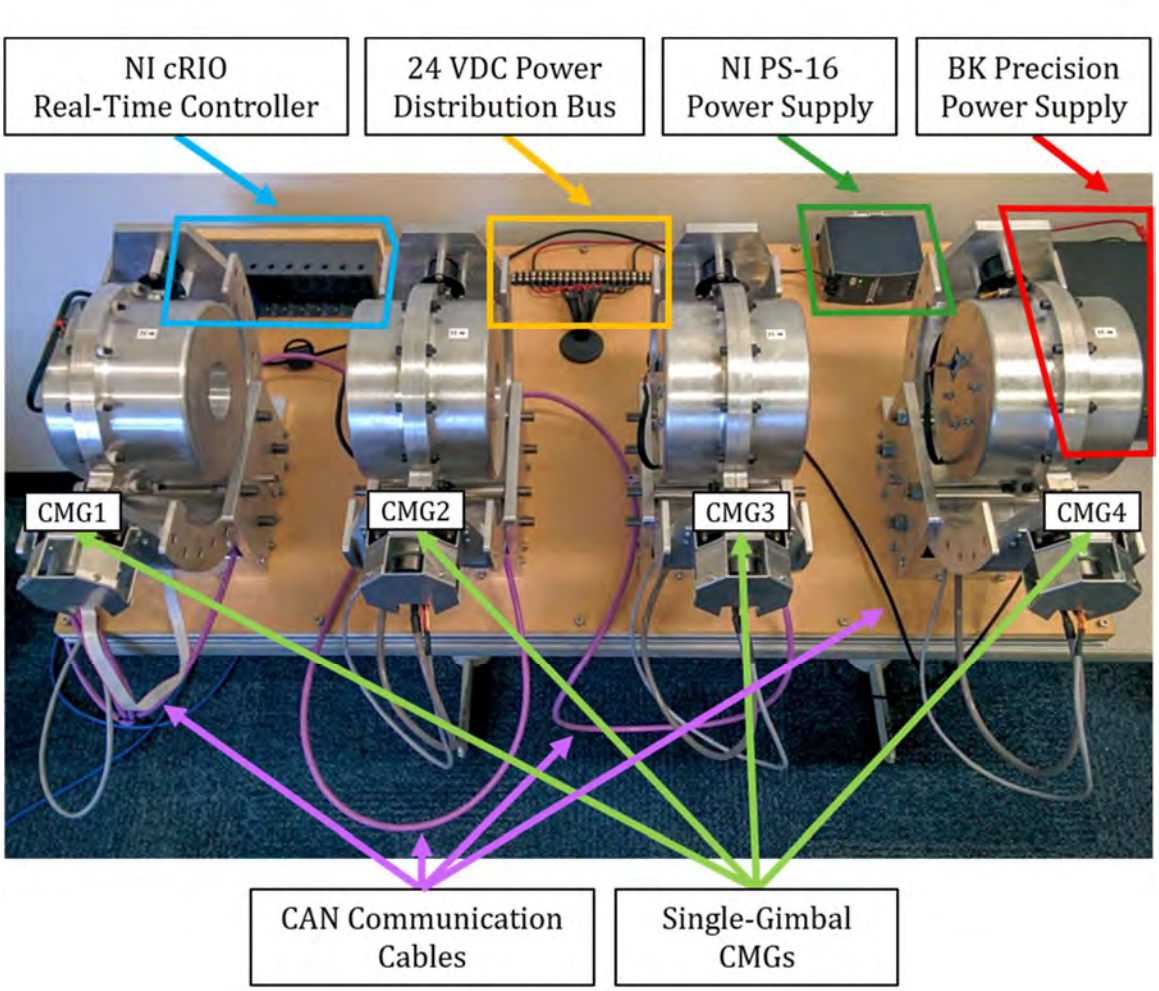

Figure 8. HIL testbed layout

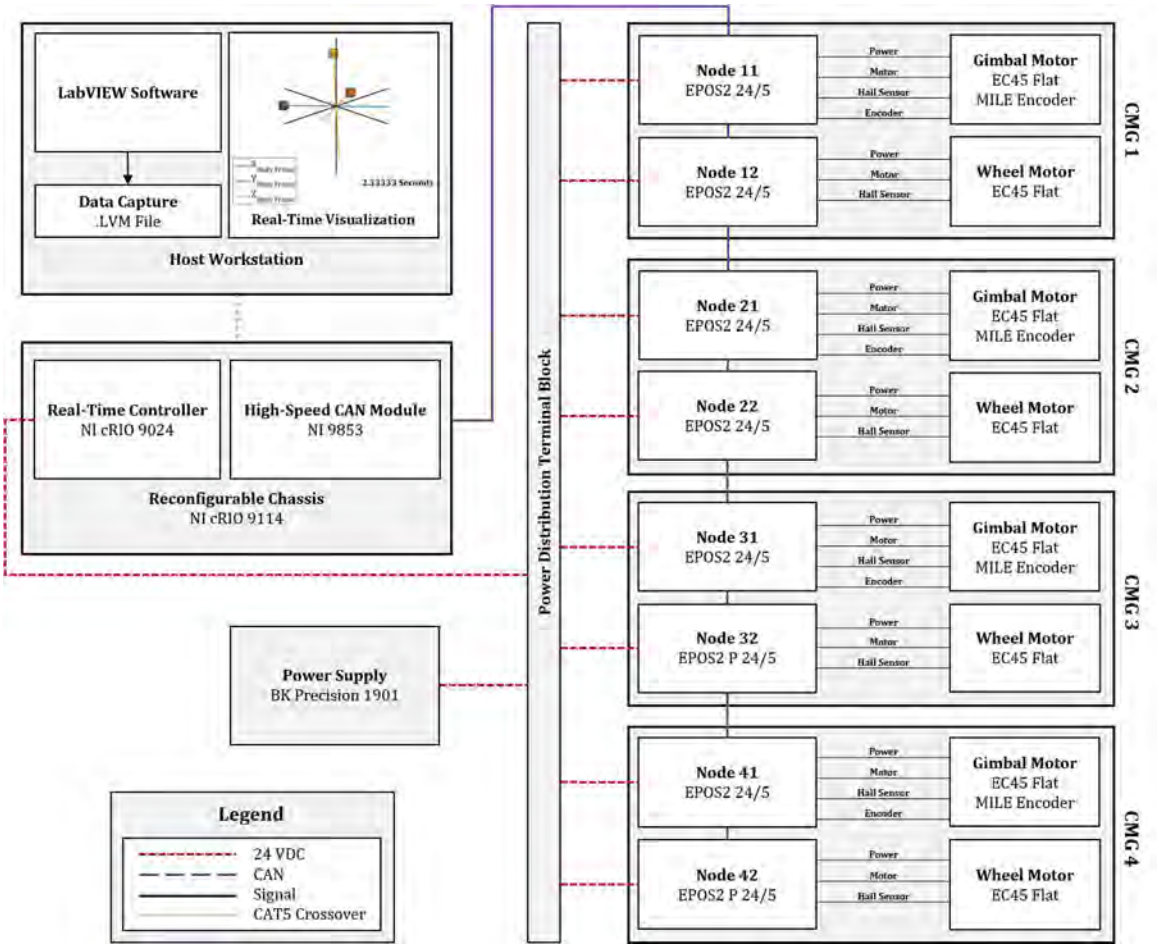

Figure 9. HIL testbed signal-flow diagram

B. OVERVIEW OF CMG MECHANISM

Each CMG is composed of an enclosed brass momentum wheel attached to a gimbal shaft. The previously-designed momentum wheels [20] are integrated onto stainless steel drive shafts and utilize a single Timken 2MM200WI precision duplex bearing set positioned near the center of mass of the wheel (see Figure 10).

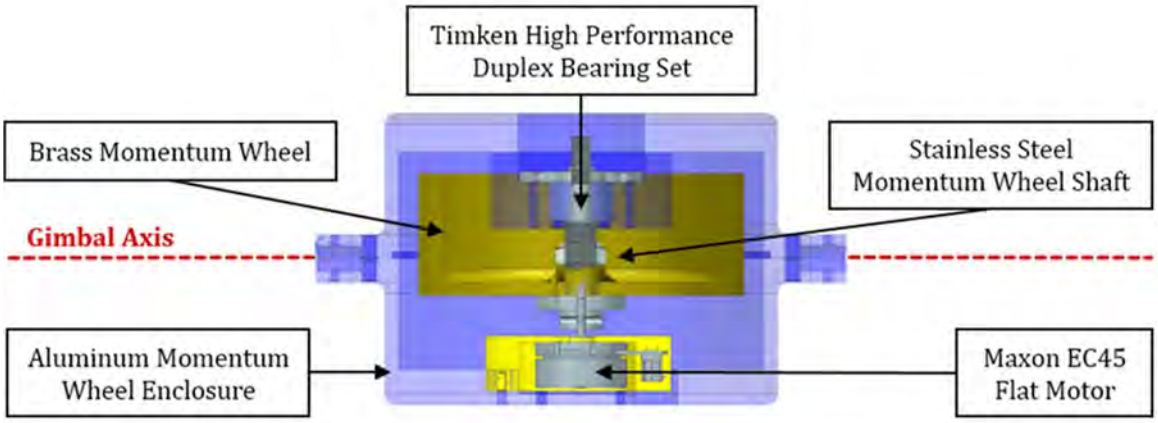

Figure 10. Section view of CMG momentum wheel enclosure

The rotational inertia of each brass momentum wheel is $0.0105\,\mathrm{kg\cdot m^2}$ and the maximum nominal angular velocity of each momentum wheel is 5000 RPM (680.1 rad/sec). This provides a nominal angular momentum magnitude of $5.5\,\mathrm{N\cdot m\cdot s}$ and a maximum torque capability of $5.5\,\mathrm{N\cdot m}$ per CMG (assuming the typical gimbal rate limit of 1 rad/sec is applied). The CMG gimbal mechanism has been designed to allow the skew angle, β, to be quickly reconfigured to any value between 45° and 90°. Standard angles (54.73° and 90°) are indexed using two quick-release pins. See [20] and [21] for a detailed discussion of the iterative design process and component selection with respect to the existing single-gimbal CMG design.

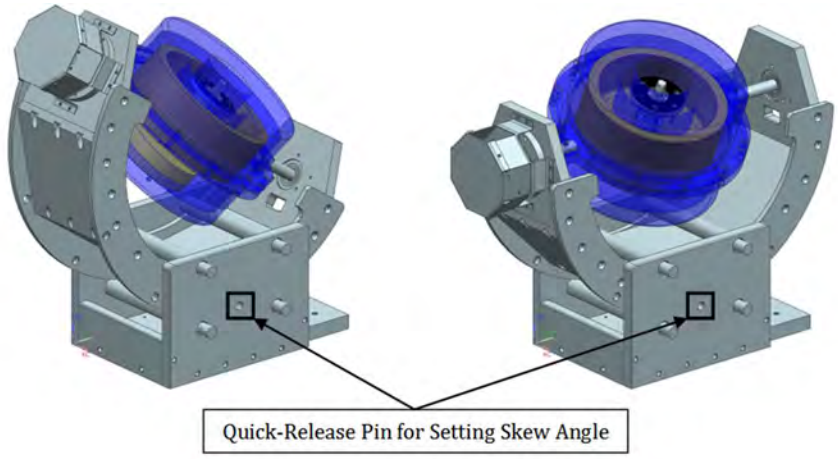

Figure 11. Rapid reconfiguration of CMG skew angle using quick-release pins

Both the momentum wheel and gimbal axis are driven by Maxon EC 45 Flat brushless DC motors. The gimbal motor includes an additional 1024-count, two-channel MILE encoder for improved control of the gimbal rate. The gimbal motor drives the gimbal shaft by means of a Harmonic Drive CSG-17-100-2UH 100:1 reduction gear. The inclusion of this reduction gear provides the gimbal motor with a significant mechanical advantage and will help to reduce unwanted gimbal back-drive effects when the CMG hardware is eventually mounted on the R-SAT three-axis simulator [21]. Power and command signals are passed to the momentum wheel motor by means of a Moog SRA-73683 12-wire slip ring. Both the momentum wheel and gimbal drives are controlled by individual Maxon EPOS2 24/5 motor controllers connected via a Controller Area Network (CAN) bus. The use of the CAN bus allows all eight motor controllers (two for each CMG) to be daisy-chained together to simplify the interface with the embedded flight computer.

C. MODIFICATIONS TO EXISTING CMG DESIGN

Several modifications were made to the CMG prototype design before it replicated for the HIL testbed. These modifications were made based on shortcomings observed during previous testing or to ease the manufacturing and assembly process for production.

1. Momentum Wheel Shaft

The previous momentum wheel shaft design was composed of separately machined shaft and motor connector components, as depicted in Figure 12. Once the brass momentum wheel was mounted to the stainless steel shaft, the shaft and motor connector were coupled using a hex key design [20]. Early testing with this design resulted in an audible chattering while accelerating or decelerating the momentum wheel, particularly at low angular velocities (less than 1000 RPM). This chattering was attributed to undesirable play in the hex key coupling between the momentum wheel shaft and the motor connector.

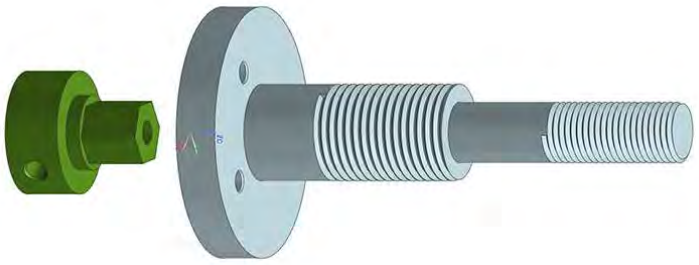

Figure 12. Previous two-piece momentum wheel shaft design

Adapted from [20]: K. L. Ackman, "Prototyping of an Open-Architecture CMG System," M.S. thesis, Dept. of MAE, Naval Postgraduate School, Monterey, CA, 2012.

In an effort to eliminate the chattering, the momentum wheel shaft and motor connector were redesigned as single component (see Figure 13). Furthermore, the outer diameter of the base of the momentum wheel shaft (formerly the motor connector) was reduced to 0.75 inches (19.05 mm) and the top of the momentum wheel shaft was machined smooth to a diameter of 0.312 inches (7.925 mm). These modifications were made to facilitate dynamic balancing of the momentum wheel assemblies.

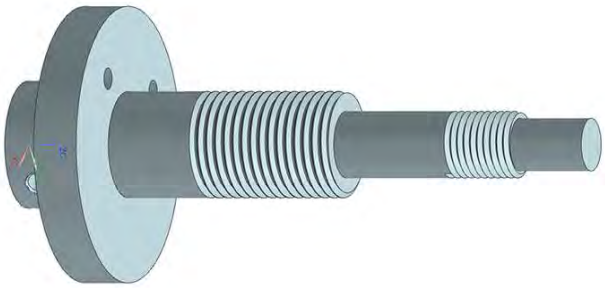

Figure 13. Updated single-piece momentum wheel shaft design

2. Momentum Wheel Case

Previous analysis of the gimbal drive noted a sinusoidal relationship between gimbal motor power consumption and the gimbal position [21]. This relationship was determined to be the result of an undesirable torque caused by a mass imbalance in the existing momentum wheel case design. It was proposed to add balance masses to the

existing momentum wheel case to correct the problem [21]. However, it was determined to be more desirable to redesign both pieces of the momentum wheel case to better balance the momentum wheel assembly prior to manufacture. Figure 14 compares the previous (left) and updated (right) momentum wheel case designs. The red and orange lines in Figure 14 illustrate the offset between the center of mass and gimbal axis, respectively. The center of mass of the momentum wheel enclosure was determined using the *Measure Bodies* tool in the Siemens *NX 9* CAD software suite. Prior to the redesign, the center of mass of the momentum wheel enclosure was located 19.934 mm from the gimbal axis of rotation. Upon redesign, the center of mass offset was reduced to 1.524 mm. The impact of this redesign on gimbal motor power consumption is discussed further in Chapter V.

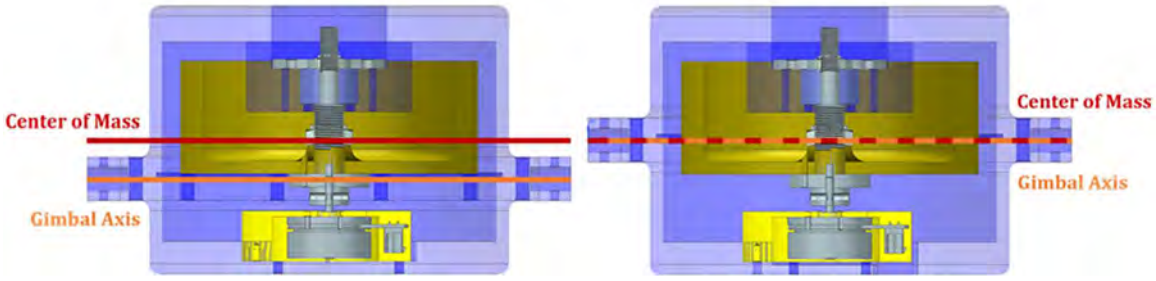

Figure 14. Side-by-side comparison of previous and updated momentum wheel case designs

D. POWER DISTRIBUTION

All eight motor controllers (two per CMG) operate on a 24 VDC bus powered by a BK Precision 1901 Switching Mode Power Supply (see Figure 15) with a maximum current output of 30 amperes (A). Power is distributed to all devices using a 20-circuit terminal block (see Figure 16). Two eight-circuit jumpers are used to wire the positive and negative leads of each motor controller in parallel. This configuration also allows for the eventual inclusion of the NI cRIO 9024 real-time controller on the main power distribution bus (the real-time controller is currently powered using a dedicated power supply).

Figure 15. BK Precision 1901 Switching Mode Power Supply

Figure 16. Power distribution terminal block

E. CONTROLLER AREA NETWORK (CAN) BUS ARCHITECTURE

Communication between the embedded flight computer and the CMG motor controllers is accomplished via a Controller Area Network (CAN) bus. The CAN architecture was first developed in 1985 as a replacement for point-to-point vehicle wiring [25]. With the increase in vehicle electronics, point-to-point wiring was becoming bulky and complex. Rather than Electronic Control Units (ECUs) communicating with each device in the vehicle individually, the CAN architecture allowed for all devices to be connected on a common bus. The ECU was fitted with a CAN transceiver which broadcast all messages to the bus; each device was then responsible for determining if the message required action or could simply be ignored. The CAN architecture was

eventually standardized as ISO 11898 and became increasingly adopted for real-time industrial control applications due to its relatively low cost, reduced complexity, and fault tolerance [26].

Data is passed across the CAN bus in 8 byte CAN frames. Networks and devices meeting high-speed CAN specifications can achieve a maximum data rate of 1 Mbps (for transmission lines less than 40 meters) [25]. The availability of CANOpen and EPOS libraries for *LabVIEW* allowed the time spent configuring the CAN communications interface with the flight computer control software to be minimized. This made the CAN architecture ideal for the laboratory environment.

F. HOST WORKSTATION

A Dell OptiPlex 7010 desktop computer serves as the host workstation for the HIL testbed. All flight computer control software was developed using the National Instruments (NI) *LabVIEW* software suite running on the host workstation (the software design process is described in Chapter IV). The embedded flight computer and host workstation communicate directly using an Ethernet crossover cable. The host workstation also includes MathWorks *MATLAB* software for plotting and analysis of simulation results outside of the real-time environment.

G. NI CRIO 9024 REAL-TIME CONTROLLER

An NI CompactRIO (cRIO) 9024 real-time controller serves as the embedded flight computer for the HIL simulation testbed (see Figure 17). The cRIO 9024 utilizes an 800 MHz Freescale processor for running real-time, deterministic applications and control systems. The controller also includes 512 MB of random access memory (RAM), 4 GB of nonvolatile memory for deployed resources and local data capture, as well as two Ethernet ports for networked communication and data transfer [27]. The defining feature of the cRIO 9024 is the Wind River *VxWorks* real-time operating system (RTOS). Real-time operating systems are designed to execute a single process with extremely precise timing (as opposed to general purpose operating systems such as Microsoft *Windows* or Apple *OS X* which experience non-deterministic performance) [28]. Real-time applications developed using the NI *LabVIEW* software suite can be deployed to the

cRIO 9024 and executed with near-constant timing between successive iterations (a critical requirement for achieving consistent and repeatable results from a real-time control system). Moreover, this arrangement mimics the behavior of a real spacecraft control subsystem, which is also designed to run in real time.

Figure 17 includes all components of the CompactRIO real-time controller system acting as an embedded flight computer. The blue cable is an Ethernet crossover cable used for direct network communication between the host workstation and the real-time target (including deployment of resources and near-real-time data capture). The purple cable is a Maxon Motor EPOS CAN-COM cable (Maxon Motor part number 275908) used for communication between the NI 9853 CAN module (CAN Master) and the first motor controller on the CAN bus (Node 11). All remaining CAN cables for communication between nodes were custom fabricated. This ensured that the overall length of the CAN transmission line remained less than 40 meters (in order to achieve a maximum 1 Mbps data rate). See [29] and [30] for a complete list of cable specifications for the CAN architecture.

The cRIO 9024 requires between 6 and 35 VDC for normal operation (a minimum of 9 VDC are required for startup) [27]. During initial testing, the controller was powered independently using an NI PS-16 24 VDC power supply (shown in Figure 18). This allowed motor controller power to be secured without impacting the embedded flight computer. Nevertheless, the cRIO 9024 is fully capable of operating on the 24 VDC main power bus, which is the preferred setup for implementation on the NPS R-SAT simulator.

Figure 17. NI cRIO 9024 Real-Time Controller, cRIO 9114 Reconfigurable Chassis, and 9853 2 Port, High-Speed CAN Module

Figure 18. NI PS-16 Power Supply

H. NI CRIO 9114 RECONFIGURABLE CHASSIS

The cRIO 9024 real-time controller is paired with an NI cRIO 9114 reconfigurable chassis. This chassis contains a Xilinx Virtex-5 LX50 reconfigurable field programmable gate array (FPGA) core and allows for the installation of up to eight C Series cRIO input/output (I/O) modules [31]. The FPGA allows low-level control

functions to implemented using hardware (logic blocks). This results in extremely accurate and repeatable performance and low latency communication with the I/O modules [28] (such as the NI 9853 used in this thesis). Existing *LabVIEW* FGPA logic was used to generate the software reference required to interface with the NI 9853 module from within the flight computer control software.

I. NI 9853 2 PORT, HIGH-SPEED CAN MODULE

Communication between the cRIO 9024 real-time controller and the Maxon EPOS2 24/5 motor controllers is accomplished using an NI 9853 2 port, high-speed CAN module (which also acts as the CAN Master device on the CAN bus). The NI 9853 module includes two Philips SJA1000 CAN controllers and a Philips TJA 1041 high-speed CAN transceiver capable of data rates up to 1 Mbps [32]. All eight motor controllers are currently being commanded over CAN Port 1. Future work could distribute communication across both CAN ports, thereby increasing the maximum CMG control frequency with respect to the data rate limitations of the CAN bus.

J. MAXON EC 45 FLAT BRUSHLESS DC MOTORS AND MAXON EPOS2 24/5 MOTOR CONTROLLERS

Each CMG utilizes two Maxon EPOS2 24/5 motor controllers; one to control the gimbal motor and one to control the momentum wheel motor. The EPOS2 24/5 is a 5 A, 24 VDC digital positioning controller capable of a variety of control modes (such as current, position, and velocity control). Communication with the motor controllers can be achieved via USB 2.0, RS-232, or CAN [29]. USB 2.0 was utilized for initialization and configuration of the motor controllers, however, all real-time communications is performed via the CAN bus.

Each motor controller requires a unique Node ID for identification on the CAN bus. Node ID assignment is accomplished using a series of seven mechanical dip switches (allowing for Node IDs between 1 and 127) [29]. For ease of identification, a convention was developed such that the first digit of the Node ID identifies the CMG and the second digit identifies the motor as the gimbal or momentum wheel (using decimals 1

and 2, respectively). Figure 19 illustrates the Node ID naming convention and associated dip switch settings for each CMG in the array.

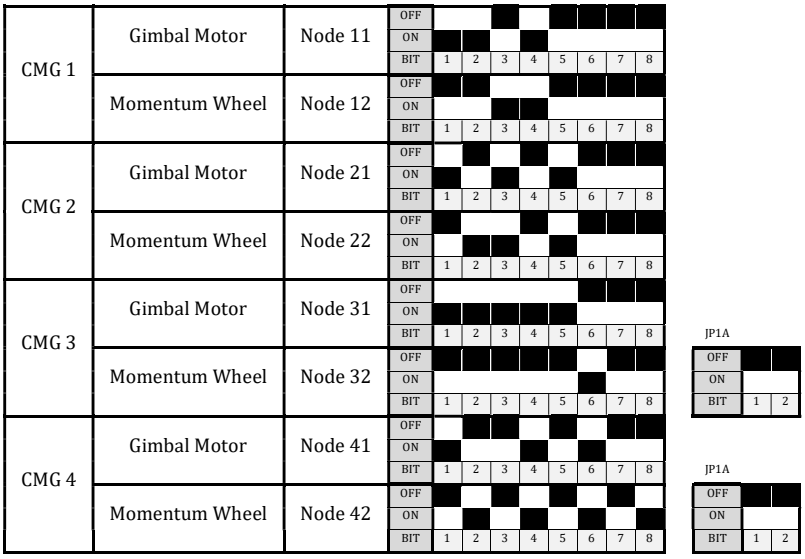

Figure 19. Motor controller Node ID assignment and dip switch settings

Two of the motor controllers used in the HIL testbed (Nodes 32 and 42) are repurposed EPOS2 P 24/5 motor controllers. The letter P indicates that these motor controllers include an additional programmable logic controller (PLC), which can serve as a CAN Master and control other devices on the CAN bus. The PLC modules (outlined in red in Figure 20) were removed from the motor controllers to enable the devices to behave as standard EPOS2 24/5 motor controllers. Due to their enhanced functionality, the EPOS2 P 24/5 motor controllers include two additional dip switches (JP1A, outlined in yellow in Figure 20) [33]; both of these dip switches must be set to the OFF position so that the motor controllers operate properly in the control loop. Additionally, the two CAN ports (CAN-S J7 and CAN-M J8) are not internally wired as is the case with standard EPOS2 24/5 motor controllers [33]. As a result, Node 32 required a custom-fabricated Y cable (connected to CAN-S J7, outlined in green in Figure 20) to allow for communication with Nodes 31 and 41. Node 42 does not required a Y cable since it is the final motor controller on the CAN bus. Finally, the bit 8 dip switch on Node 42 is set to

the ON position to activate an internal 120 ohm resistor required for CAN bus termination [33].

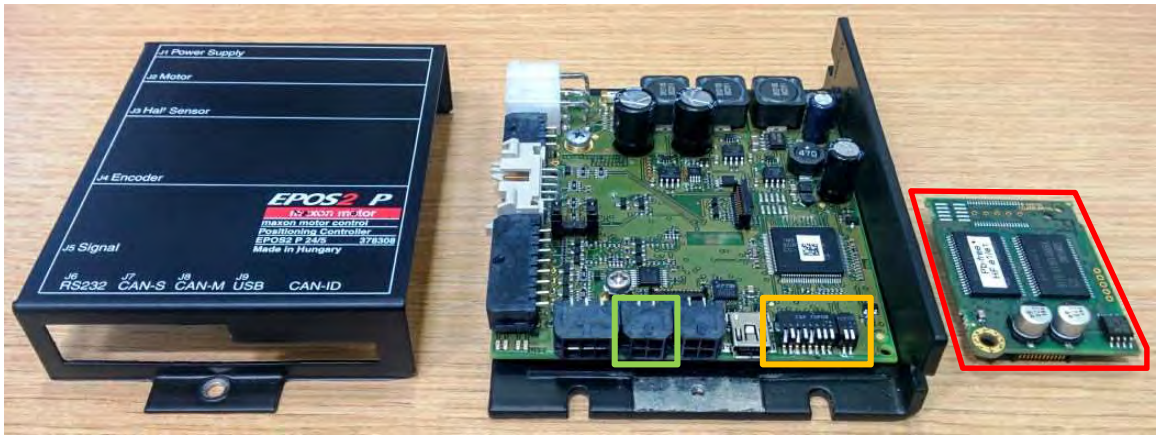

Figure 20. Maxon EPOS2 P 24/5 motor controller with PLC module removed

Motor controller initialization and tuning was performed using Maxon *EPOS Studio* software running on a Microsoft *Windows* workstation. Motor controller initialization was performed using the Setup Wizard tool in *EPOS Studio*; see Appendix A for a complete listing of the Setup Wizard settings used for the gimbal and momentum wheel motor controllers. The procedure outlined in Appendix A should be followed if any of the motor controllers are replaced in the future.

For the HIL testbed, the EPOS2 24/5 motor controllers are operated in velocity mode, which utilizes an internal 1 kHz PI controller [34] to maintain the nominal rate of the momentum wheel or to track the commanded gimbal rate. Figure 21 provides an overview of the controller architecture, wherein the PI controller commands a current set point in order to actuate the motor.

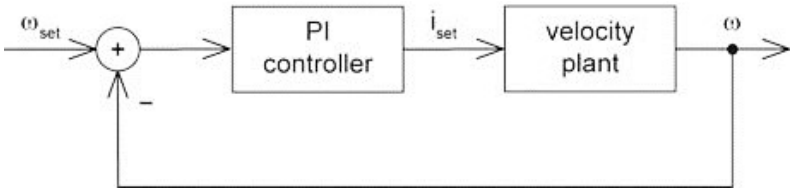

Figure 21. Maxon EPOS2 24/5 internal controller (velocity mode)

Adapted from [34]: *EPOS2 Application Notes Collection*, Maxon Motor AG, Sachseln, Switzerland, 2014.

The internal controller gains (K_P and K_I) can be set manually via the motor controller Object Dictionary or automatically using the Regulation Tuning tool in *EPOS Studio*. The controller gains were determined using the Regulation Tuning tool. Since each motor controller was configured individually, the internal controller gains set using the Regulation Tuning tool varied slightly between each device. In order to standardize the response of all four CMGs, the average values of the automatic controller gains were calculated and then applied to the motor controllers using the EPOS Object Dictionary (this process was performed separately for the gimbal and momentum wheel motor controllers). Eqns. (28) and (29) express the conversion from EPOS Studio gain values to gain values in engineering units. Table 1 summarizes the internal motor controller gains. Future work could be performed to optimize the controller gains to achieve a specific system response or to mimic the characteristics of a particular CMG.

$$K_P\left(\text{Engineering Units}\right) = 20\frac{\mu\text{A}}{\text{rad/sec}} \times K_P\left(\text{EPOS Units}\right) \tag{28}$$

$$K_I\left(\text{Engineering Units}\right) = 5\frac{\text{mA/sec}}{\text{rad/sec}} \times K_I\left(\text{EPOS Units}\right) \tag{29}$$

Table 1. Maxon EPOS2 24/5 internal controller gains (velocity mode)

Node ID	K_P (EPOS Units)	K_I (EPOS Units)	K_P (Eng. Units)	K_I (Eng. Units)
Node 11	3423	870	68460	4250
Node 12	1537	82	30740	410
Node 21	2048	234	40960	1170
Node 22	1789	95	35780	475
Node 31	2067	226	41340	1130
Node 32	1584	89	31680	445
Node 41	2100	228	42000	1140
Node 42	1226	67	24520	335

Motor Controller Mean	K_P (EPOS Units)	K_I (EPOS Units)	K_P (Eng. Units)	K_I (Eng. Units)
Gimbal	2410	390	48200	7800
Momentum Wheel	1534	83	30680	1660

K. SUMMARY

This chapter provided an overview of the CMG array and HIL testbed architecture. The individual components were described along with the communication and power architecture required to command the system in real time. Chapter IV will discuss the development of the embedded flight computer control software and the way in which the software communicates with the physical elements of the HIL testbed.

IV. FLIGHT COMPUTER CONTROL SOFTWARE DESIGN

This chapter provides an overview of the *LabVIEW* software suite and describes the development of the control software implemented on the embedded flight computer to simulate the spacecraft attitude dynamics and perform real-time attitude control and CMG steering. While it is possible to integrate existing *MATLAB* code or *Simulink* models with *LabVIEW*, the goal of this thesis was to develop the entire attitude control law, CMG steering law, and simulated spacecraft dynamics model natively in *LabVIEW* in order to maximize the real-time performance of the system. Although the physical arrangement of the CMGs during an HIL simulation is inconsequential, the CMG steering law has been developed to support the standard pyramidal configuration described in Chapter II (where the gimbal axis of CMG1 is aligned with the $+\hat{X}$ axis when the skew angle is set to 90°). Figure 22 provides a top-down view of the existing NPS R-SAT three-axis simulator along with the corresponding location of the four replacement CMGs. HIL simulation of other configurations is made possible simply by adjusting trigonometric relationships in flight computer control software (and the spacecraft dynamics model).

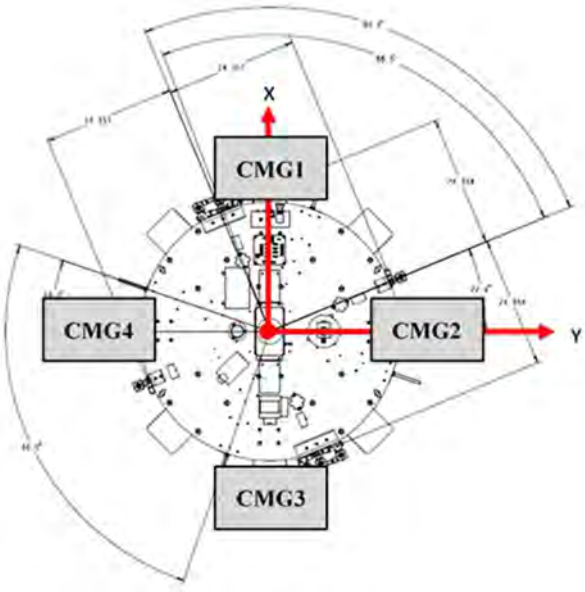

Figure 22. Schematic top view of NPS R-SAT simulator attitude stage

Adapted from [19]: *3DOF Satellite Simulator User's Guide*, Andrews Space, Tukwila, WA, 2009.

A. LABVIEW SYSTEM DESIGN SOFTWARE

LabVIEW is an industry-standard software suite for developing instrument control and data acquisition applications, otherwise known as Virtual Instruments (VIs). VIs are developed using a graphical programming language known as *G* and compiled into machine code. Once compiled, VIs can be deployed to a variety of target devices, including real-time controllers (such as the cRIO embedded computer). *LabVIEW* includes a variety of modules for detailed application design (such as *LabVIEW* Real Time, which is designed specifically for applications running on real-time targets, and *LabVIEW* FPGA, which can be used to develop logic for implementation on an FPGA (such as the Xilinx Virtix-5 FPGA on the cRIO 9114 backplane).

LabVIEW VIs consist of interactive Front Panels and underlying Block Diagrams. The VI Front Panel allows the user to interface with the application and its associated hardware and can include various controls (user inputs) and indicators (outputs) as well as advanced plotting and visualization tools. The VI Block Diagram contains the underlying logic which dictates the behavior of the application and the devices it interfaces with [35]. *LabVIEW* (and the NI cRIO family of devices) are ideal candidates for the HIL testbed because of the way in which they simplify the process of collecting and analyzing data and commanding external devices.

LabVIEW Block Diagrams appear similar to functional wire diagrams and are based on the concept of dataflow [35]. Dataflow means that functions and subroutines only execute once they have received all necessary inputs. This behavior implies that while ***Timed Loops***[5] can be used to impose a desired control frequency, the various elements of the control logic will not execute until all inputs have been updated for that specific iteration. At a most basic level, *LabVIEW* Block Diagrams consist of three elements: terminals, nodes, and wires. Every control and indicator on a VI Front Panel has an associated terminal on the Block Diagram. Dataflow between controls and indicators is dictated by means of wires. The color and thickness of a wire indicates the data type and structure of the signal. Nodes are anything which manipulate a signal.

[5] For the remainder of this thesis, bold italics will be used to denote structures in *LabVIEW*.

Structures are used to impose specific functionality on a group of logic. Examples of structures used in the design of the embedded of flight computer software are ***Timed Loops*** (which iterate with a specific loop duration), ***For Loops*** (which iterate a specific number of times), ***While Loops*** (which iterate until a specific stop condition is met), and ***Case Structures*** (which perform different logic depending on the state of a Boolean trigger). Signals are typically passed into and out of structures using one of two methods; tunnels and shift registers. Tunnels pass the signal value at the start of structure execution while shift registers pass signals between successive loop iterations. Local Variables were used extensively during software design to pass signals into and out of parallel structures. In order to avoid what *LabVIEW* defines as a "race condition," it was critical to ensure that a Local Variable could only be written to in one location (a Local Variable can be read in multiple locations without risking a race condition). As previously mentioned, dataflow requires that a node receive all required inputs prior to executing. Since the spacecraft dynamics model and attitude control law was implemented in a 100 Hz ***Timed Loop*** and the CMGs are only being commanded and sampled at a frequency of 10 Hz, there are certain Local Variables (notably the momentum wheel angular velocities and gimbal positions) which are being read more frequently than they are written to. The behavior of Local Variables is such that they maintain their previous value until updated. This results in zero-order sample and hold behavior (meaning that the system will continue to report the same momentum wheel angular velocities and gimbal positions for multiple iterations of the spacecraft dynamics and attitude control loop) despite the fact that the actual response of the physical hardware is varying in continuous time. This is one reason why an HIL simulation cannot fully replace testing with a three-axis air bearing test stand.

Many of the *LabVIEW* libraries include existing functions and subroutines which simplified the process of implementing the spacecraft dynamics model, quaternion feedback control law, and CMG steering. Maxon provides an EPOS Instrument Driver which utilizes functionality from the *LabVIEW* CANopen Library to simplify the process of commanding EPOS motor controllers using a cRIO and NI CAN module.

Multiple VIs can be organized in a *LabVIEW* Project, which includes references to any resources (such as *LabVIEW* libraries) required for program execution (these items are included under Dependencies in the *LabVIEW* Project structure). When necessary, these resources can be deployed to target devices for execution. Furthermore, the *LabVIEW* Project structure allows users to select which VIs to run. This functionality is particularly useful since it allows for multiple versions of a VI to be deployed to a target device (each with different parameters) and run individually. For the purposes of HIL testing, multiple VIs can be generated based on the developed flight computer control software, each with the representative mass properties and control system limitations of a respective spacecraft. Rather than requiring the user to modify these parameters prior to an HIL simulation (some of which are hard-coded into the VI Block Diagram), the user can simply select the VI associated with the specific spacecraft they would like to evaluate the maneuver on. Figure 23 provides an overview of the developed *CMG Flight Computer.lvproj* Project. This *LabVIEW* Project contains the two real-time applications, *CMG Flight Computer.vi* and *Reset Gimbal Positions.vi*, as well as the soft real-time application (*Reader VI.vi*) which runs on the host workstation for data capture and attitude visualization.

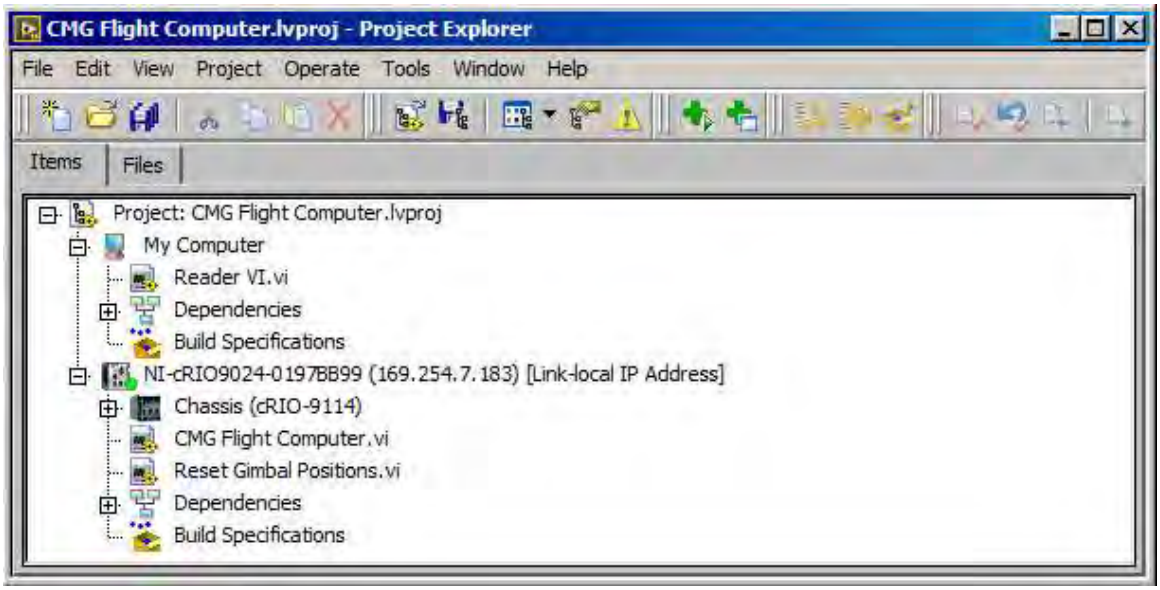

Figure 23. CMG Flight Computer.lvproj Project Explorer

B. FLIGHT COMPUTER CONTROL SOFTWARE OVERVIEW

The *CMG Flight Computer.vi* is based upon two real-time loops. The first ***Timed Loop*** runs at 100 Hz and implements the attitude control law, the CMG steering law, and the simulated spacecraft dynamics model. The second ***Timed Loop*** operates at 10 Hz and is responsible for sending and receiving the CAN Process Data Objects (PDOs) used for commanding the gimbal and momentum wheel motors and receiving system telemetry. A block diagram of the overall HIL architecture is shown in Figure 24, which shows the functions performed within each ***Timed Loop*** as well as the partition between the simulated world and physical hardware.

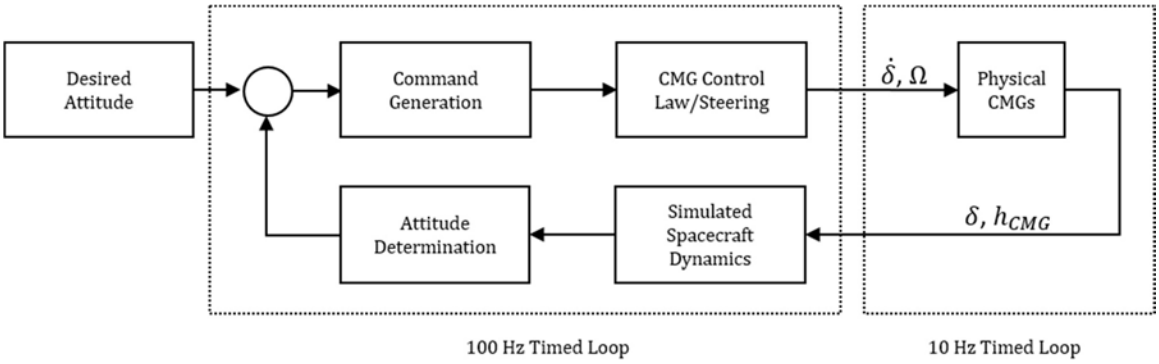

Figure 24. Flight computer conceptual block diagram

C. SIMULATION CONTROL

Control of the simulation is performed from the developed *CMG Flight Computer.vi* Front Panel (see Figure 25). This user interface (UI) allows for control of the initial, $\bar{q}_0$, and desired, $\bar{q}_c$, quaternion, quaternion feedback control gains, k and c, skew angle, β, and gimbal rate limit, $\dot{\delta}_{max}$. Once the VI is running, the UI also allows for enabling/disabling the eight motor controllers, setting the desired momentum wheel angular velocity, Ω_{CMG}, and starting/stopping the simulation. Additional controls have been included to allow for manual control of the gimbal rate (outside of the attitude control simulation environment) to allow for other system characterization tests to be performed.

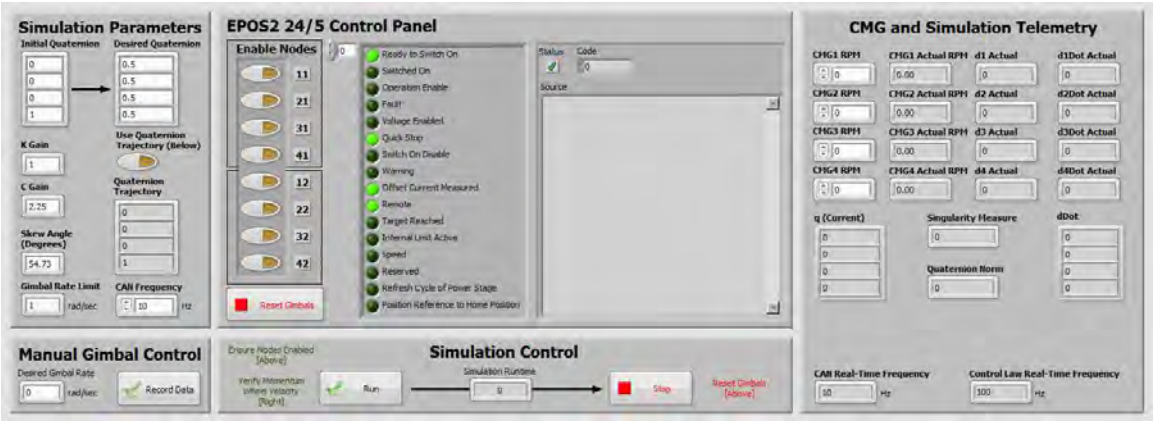

Figure 25. CMG Flight Computer.vi Front Panel

D. QUATERNION FEEDBACK CONTROL LAW

As discussed in Chapter II, quaternion feedback is used as the attitude control solution for the HIL testbed. Alternate control solutions can be tested by simply updating the flight software. The implementation of quaternion feedback control in real-time logic is now presented.

(1) Commanded Quaternion

The flight computer control software is designed to accept either a static quaternion (step input) or quaternion trajectory (generated outside of the real-time environment) as the commanded quaternion. The commanded quaternion matrix (Q) described in Eqn. (18) is updated at a frequency of 10 Hz using a 100 ms ***Timed Loop*** in *LabVIEW* (see Figure 26). The value of the commanded quaternion matrix is constant when using a static quaternion input. In the event a quaternion trajectory is utilized, the control software selects the appropriate quaternion sample, $\vec{q}_c(t_i)$, from the quaternion trajectory based on the current iteration, i, of the ***Timed Loop***. The software also contains logic to automatically stop the simulation and reset the gimbal angles once the end of a quaternion trajectory is reached. Commanded quaternion input selection (static or trajectory) is performed using a Boolean control on the VI Front Panel.

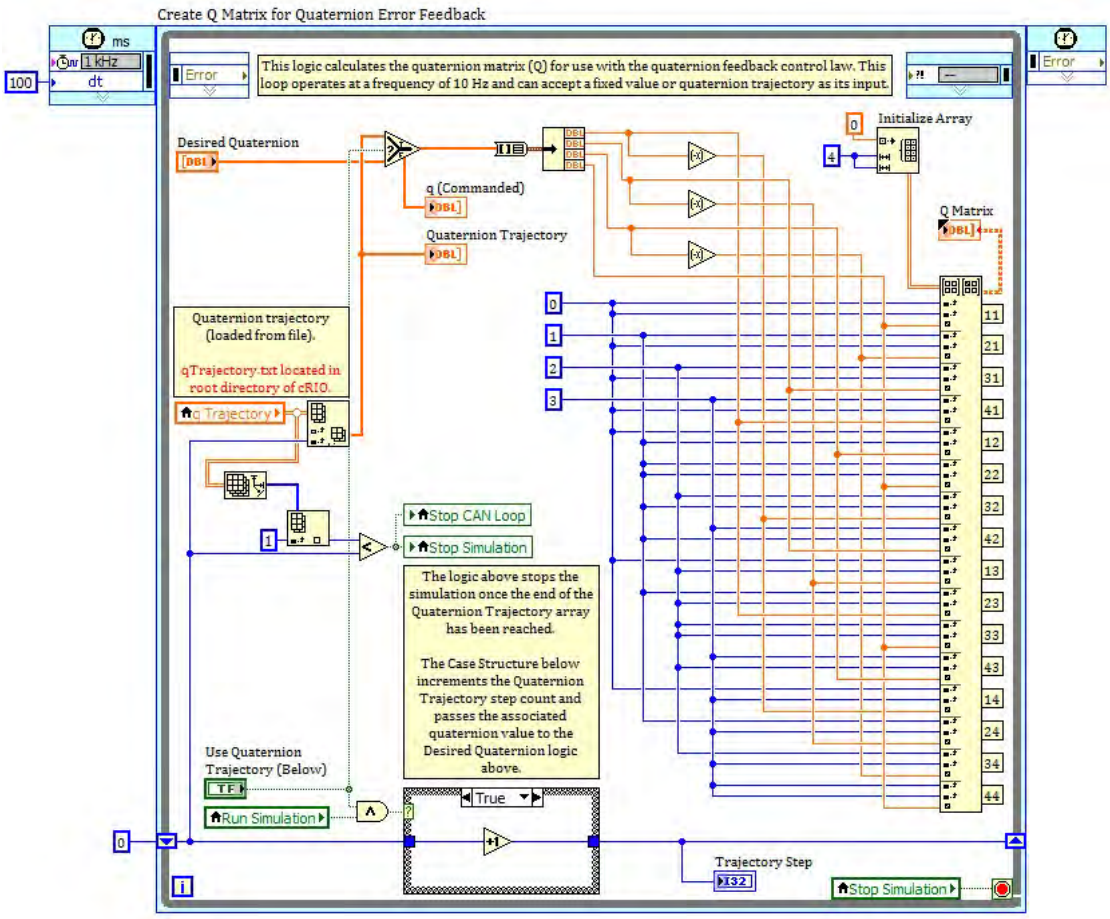

Figure 26. Commanded quaternion selection and Q matrix generation in *CMG Flight Computer.vi*

(2) Quaternion Error Computation

The attitude error quaternion vector, $\bar{q}_e$, used in the quaternion feedback control law (Eqn. (19)) is calculated using Eqn. (18). The control law only requires the first three elements of the attitude error quaternion vector, therefore the *Delete from Array* function is used to remove the final element, q_{4e}. Since *LabVIEW* utilizes zero-based indexing, the final element of the attitude error quaternion vector is indexed as element 3 (as depicted in Figure 27).

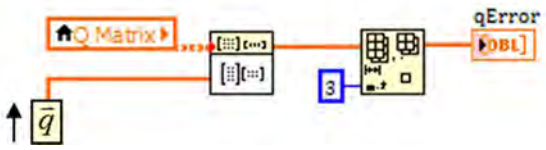

Figure 27. Attitude error quaternion calculation in *CMG Flight Computer.vi*

(3) Control Torque

The control torque, $\bar{u}$, is calculated as described in Eqn. (19) and as depicted in Figure 28. The default gain values k and c can be easily modified using an input on the VI Front Panel.

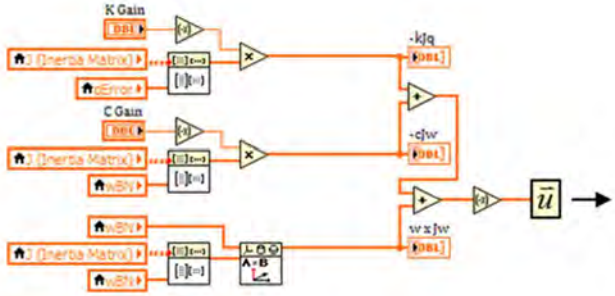

Figure 28. Control torque calculation in *CMG Flight Computer.vi*

E. CMG STEERING LAW

Despite the fact that the CMG control is carried out at 10 Hz, the CMG steering law is computed within the 100 Hz ***Timed Loop***. The required gimbal rates are saved as Local Variables; only the most recent values are used by the 10 Hz CAN bus ***Timed Loop***. The details of each step in the steering law computation are now discussed.

(1) Jacobian Matrix

The Jacobian matrix is calculated at a rate of 100 Hz using the most recent gimbal angle measurements, δ_{CMG}, and the calculated angular momentum of the CMG array, $\bar{h}$. This calculation is based on the angular rates of the momentum wheels measured from the CMG hardware. The sine and cosine of each gimbal angle is calculated once per iteration and is passed to both the CMG angular momentum and Jacobian matrix logic (to

minimize the number of trigonometric functions being calculated in real-time). The Jacobian matrix logic is depicted in Figure 29.

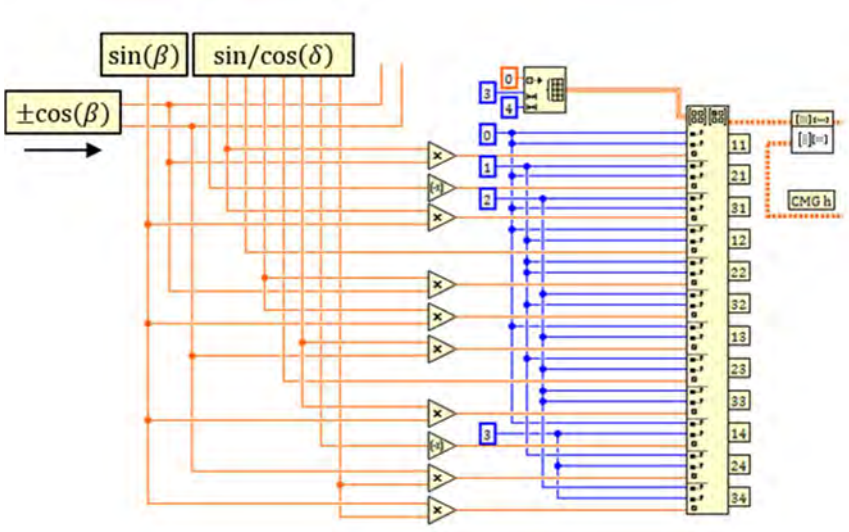

Figure 29. Jacobian matrix calculation in *CMG Flight Computer.vi*

(2) Gimbal Rate

The required gimbal rates, $\dot{\delta}_{CMG}$, are calculated next using the Moore-Penrose pseudoinverse of the Jacobian matrix and the desired control torque, $\vec{u}$, as described in Eqn. (25). The flight computer control software is designed with additional logic to limit the commanded gimbal rates to a maximum permissible gimbal rate, $\dot{\delta}_{max}$. This rate limit can be adjusted using an input on the VI Front Panel. Eqn. (30) is based on the inf-norm saturation function described in [4] and represents the gimbal saturation logic as implemented in *LabVIEW* (see Figure 30). The CMG steering law logic utilizes the inf-norm to normalize the commanded gimbal rate vector, $\dot{\vec{\delta}}$, by the maximum requested gimbal rate, then scales the entire vector by the maximum permissible gimbal rate, $\dot{\delta}_{max}$.

$$\text{sat}\left(\dot{\vec{\delta}}\right) = \begin{cases} \dot{\vec{\delta}} & \text{if } \left\|\dot{\vec{\delta}}\right\|_\infty < \dot{\delta}_{max} \\ \dot{\delta}_{max} \times \dfrac{\dot{\vec{\delta}}}{\left\|\dot{\vec{\delta}}\right\|_\infty} & \text{if } \left\|\dot{\vec{\delta}}\right\|_\infty \geq \dot{\delta}_{max} \end{cases} \tag{30}$$

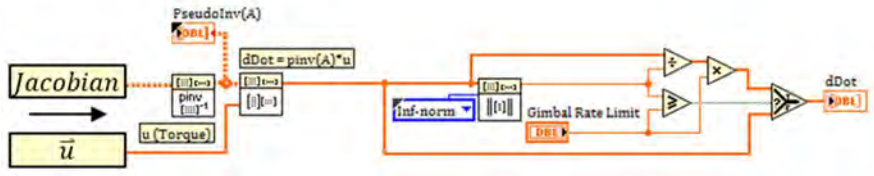

Figure 30. CMG gimbal rate calculation and saturation logic in *CMG Flight Computer.vi*

This implementation of gimbal rate saturation logic maintains the direction of the commanded torque vector while scaling the magnitude to remain within the maximum achievable torque envelope. Since the flight computer control software is utilizing quaternion feedback, the requested torque vector should be approximately collinear with the instantaneous Eigenaxis of the maneuver (with some deviation to account for steering around singular points). As was learned during development, an incorrect approach to gimbal rate saturation would simply limit each individual gimbal rate to some maximum value. In this case, the actual torque vector generated by the CMG array would differ from the commanded torque vector in both magnitude and direction.

F. SIMULATED SPACECRAFT DYNAMICS

In the HIL simulation, the attitude dynamics of the spacecraft must be simulated. This simulation was performed as part of the real-time software running on the cRIO. The current spacecraft model assumes zero external disturbance torque and zero-momentum bias. Future work could include model updates to allow for momentum bias and disturbance terms. The theory of conservation of angular momentum dictates that (based on the assumptions above) the angular momentum of the simulated spacecraft, $\vec{h}_{Body}$, is equal in magnitude and opposite in direction to the net angular momentum of the CMG array, $\vec{h}_{CMG}$. The instantaneous angular velocity of the spacecraft, ω_{BN}, can therefore be estimated based on the measurements of the current net angular momentum measured from the CMG array. Assuming a zero-bias control system, the instantaneous angular velocity of the spacecraft can be derived as follows:

$$\begin{aligned} \vec{h}_{Body} + \vec{h}_{CMG} &= 0 \\ \vec{h}_{Body} &= -\vec{h}_{CMG} \\ J\omega_{BN} &= -\vec{h}_{CMG} \\ \vec{\omega}_{BN} &= -J^{-1}\vec{h}_{CMG} \end{aligned} \tag{31}$$

The net angular momentum of the CMG array, $\vec{h}_{CMG}$, is determined at each time step using Eqn. (21) and the current measured gimbal positions, δ_{CMG}, and measured momentum wheel angular velocities, Ω_{CMG}, as reported by the motor controllers via the CAN bus. The logic used to perform this calculation is depicted in Figure 31.

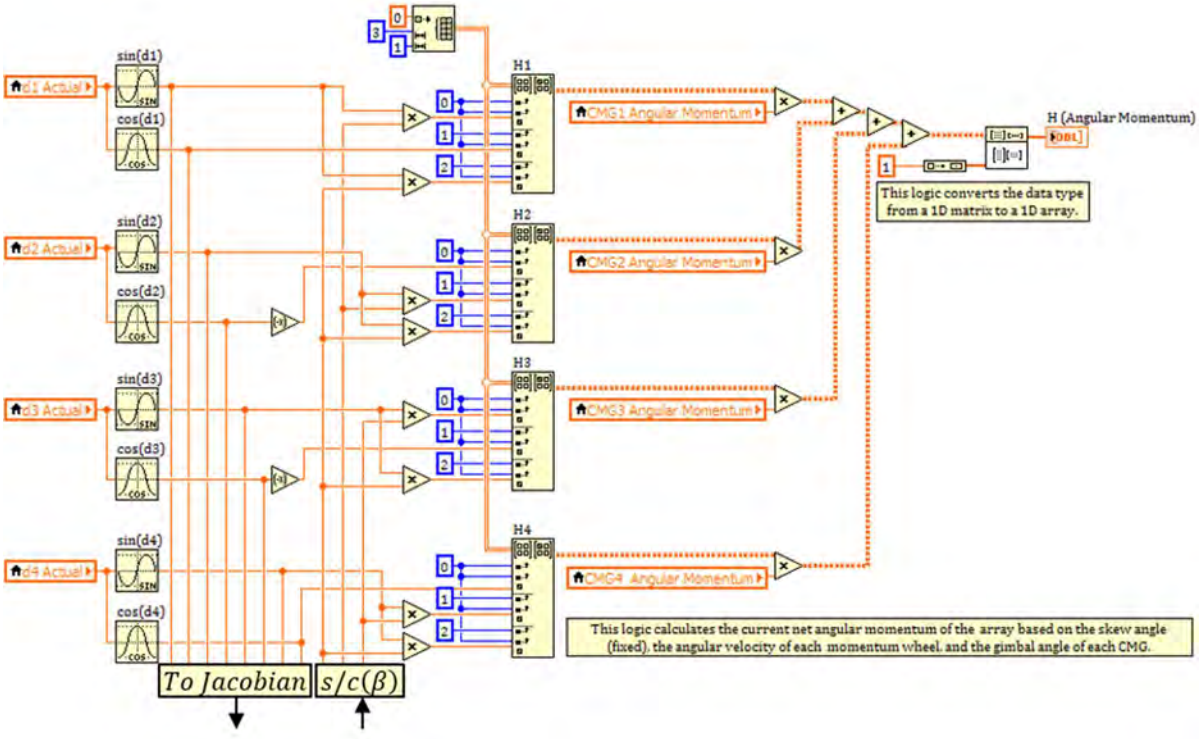

Figure 31. CMG net angular momentum calculation in *CMG Flight Computer.vi*

(1) Simulated Inertia Matrix

In the results presented here, the inertia matrix, J, of the existing NPS R-SAT simulator was used for simulating the spacecraft dynamics. The R-SAT inertia tensor is given as [19]:

$$J = \begin{bmatrix} 37.25 & 0.59 & 0.05 \\ 0.59 & 39.88 & 0.09 \\ 0.05 & 0.09 & 70.03 \end{bmatrix} \text{kg} \cdot \text{m}^2$$

The spacecraft is assumed to be a rigid body and, as such, the inertia matrix, J, is constant for the duration of the simulation. The inertia matrix, J, and its inverse, J^{-1} are generated once upon initialization of the real-time VI. Figure 32 depicts the inertia matrix generation and inverse inertia matrix generation in *LabVIEW*. Note that in Figure 32, the inertia values are given in English units (as provided in [19]) are converted to SI by a scaling factor.

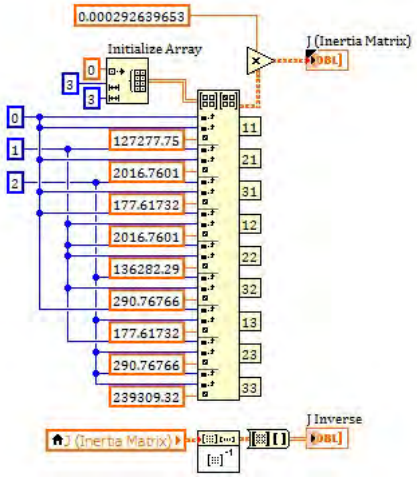

Figure 32. Inertia matrix generation in *CMG Flight Computer.vi*

(2) Quaternion Propagation

As previously discussed, the simulated spacecraft attitude is parameterized using quaternions. The instantaneous quaternion kinematics, $\dot{\bar{q}}(t)$, are calculated in real-time using Eqn. (13) and the current angular velocity as given by Eqn. (31). The spacecraft

attitude is then propagated using an Euler step with $\Delta t = 0.01$ sec. Using this simple approach, the quaternion at time t_i is calculated as:

$$\vec{q}(t_i) = \vec{q}(t_{i-1}) + \dot{\vec{q}}(t_i)\Delta t \tag{32}$$

The quaternion vector ($\vec{q}$) is normalized at each time step to conform with the constraint $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$. For ease of implementation in *LabVIEW*, the value of the quaternion norm is calculated using the square of the 2-norm. The 2-norm is defined as [24]:

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

Figure 33 depicts the implementation of the quaternion kinematics and attitude propagation in *LabVIEW*. The quaternion vector, $\vec{q}$, and angular velocity vector, ω_{BN}, calculated at time t_i are then used for attitude feedback in the spacecraft attitude control law; the current quaternion vector, $\vec{q}(t_i)$, is also passed to the next iteration of the ***Timed Loop*** (where it is stored as $\vec{q}(t_{i-1})$) using a *Shift Register*.

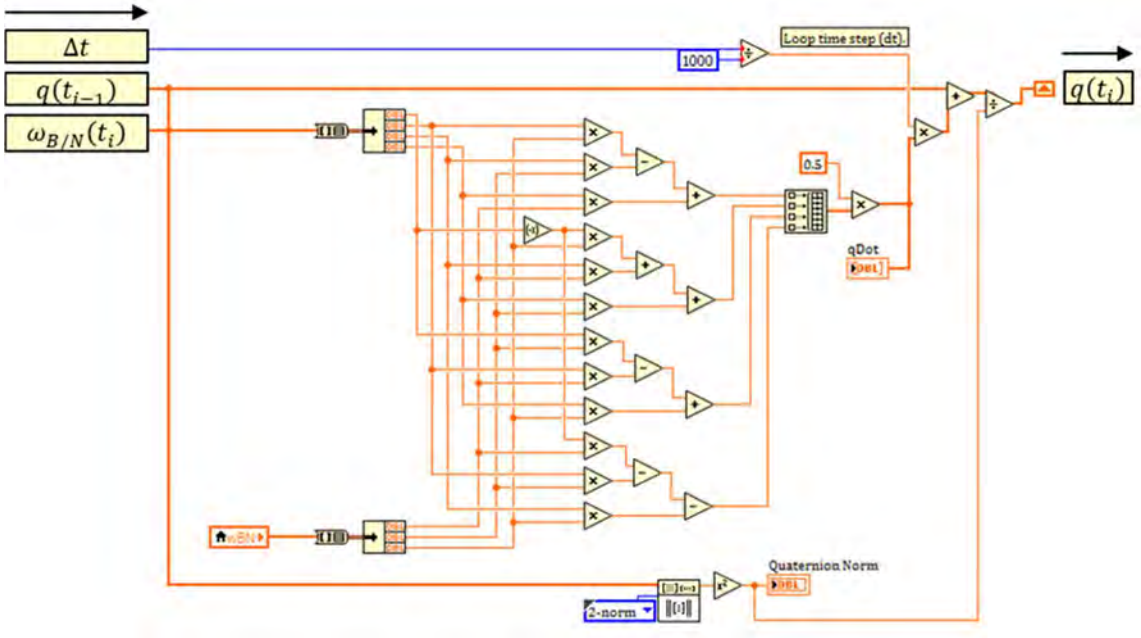

Figure 33. Quaternion kinematics and attitude propagation in *CMG Flight Computer.vi*

G. MOTOR COMMAND GENERATION AND CMG TELEMETRY

As described previously, the Maxon EPOS2 24/5 motor controllers are configured to operate in velocity mode. With this approach, attitude control functions are only performed by the control software running on the real-time controller. The logics of the individual motor controllers are assumed to be inner loops and are used to achieve and maintain the rates (Ω or $\dot{\delta}$) commanded by the higher-level attitude control system.

Initialization of the CAN bus and configuration of the motor controllers is performed each time the main *CMG Flight Computer.vi* is executed. Existing EPOS and CANopen libraries were used to minimize the amount of low-level programming required to implement a CAN communication architecture. While the maximum angular velocity of the Maxon EC 45 Flat motors is 10000 RPM, this limit is reduced to 3125 RPM when using a sinusoidal commutation command signal with an 8 pole pair motor. As such, the gimbal motor controllers and momentum wheel motor controllers must be configured separately in order to allow the momentum wheel motors to operate above 3125 RPM. A custom *EPOS2 Configure.vi* mode was created named "Velocity with Current" (see Figure 34). This mode configures the motor controllers to operate in velocity mode (accepting angular velocity as the control input) but includes additional logic to have the motor controllers report "Current Actual Value" (0x6078 in the EPOS Object Dictionary) in addition to position and velocity telemetry.

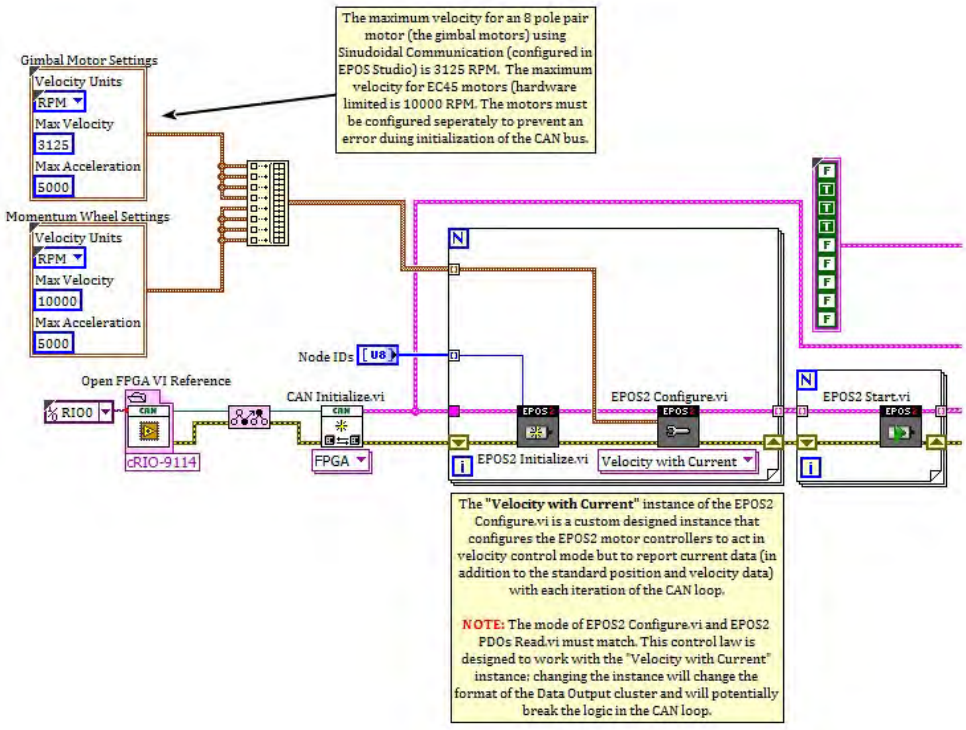

Figure 34. Maxon EPOS2 24/5 motor controller configuration logic in *CMG Flight Computer.vi*

The native unit of angular velocity for the EPOS2 motor controllers is RPM. While momentum wheel angular velocity is traditionally expressed in RPM, the CMG steering law generates required gimbal rates, $\dot{\vec{\delta}}$, expressed in rad/sec. As such, gimbal motor commands must be converted from rad/sec to RPM. Furthermore, gimbal motor commands must account for the 100:1 gear ratio of the reduction gear. Motor position is provided by the EPOS2 motor controllers in terms of encoder counts. The resolution per revolution is dependent upon the style of sensor used. As previously discussed, the momentum wheel motors utilize Hall Effect sensors with a resolution of 48 states per revolution. The gimbal motors utilize Maxon MILE encoders with a resolution of 4096 encoder counts per turn. Coupled with the reduction gear, the gimbal motors report the position of the gimbal shafts with a resolution of 409600 encoder counts per turn or 8.789×10^{-4} degrees/bit.

(1) Momentum Wheel

While the spacecraft dynamics model and CMG steering law are not concerned with the angular position of the momentum wheels (Θ), the limited resolution of the Hall Effect sensors result in inconsistent instantaneous angular velocity (Ω) estimates. As such, the difference in angular position between successive cycles of the CAN control loop is used to estimate the momentum wheel angular velocity with greater accuracy. In order to increase the accuracy of this estimation, the time step (Δt) used in Eqn. (33) is the difference in controller clock time (vice the commanded cycle period). This approach minimizes the impact of jitter on the angular velocity estimation. Eqns. (34) and (35) express the conversion from momentum wheel motor encoder count to angular position in radians.

$$\Omega_{CMG}\left(t_i\right) = \frac{d\Theta}{dt} = \frac{\Theta\left(t_i\right) - \Theta\left(t_{i-1}\right)}{t_i - t_{i-1}} \tag{33}$$

$$\Theta\left(\text{Encoder Count}\right) \times \text{Conversion Factor} = \Theta\left(\text{rad}\right) \tag{34}$$

$$\text{Conversion Factor} = \frac{1 \text{ rev}}{48 \text{ Encoder Counts}} \times \frac{2\pi \text{ rad}}{1 \text{ rev}} = 0.1309 \frac{\text{rad}}{\text{Encoder Count}} \tag{35}$$

While the current embedded flight computer control software is designed to implement fixed-speed CMG steeling law, variable-speed CMG (VSCMG) control law can be easily implemented with minimum rework. The CAN loop currently reads the user-specified control value for momentum wheel angular velocity using Local Variables. Should VSCMG control law be implemented in the attitude dynamics and control loop, the Local Variable nodes in the CAN loop simply need to be updated to read the real-time momentum wheel angular velocity command.

(2) Gimbal Rate

Unlike with the momentum wheels, the spacecraft dynamics model and CMG steering law are dependent upon both the angular position and angular velocity of the CMG gimbals. The MILE encoders used on the gimbals are more than capable of determining gimbal rate with the fidelity required for accurate spacecraft dynamics

simulation. As previously discussed, the EPOS2 24/5 motors controllers work in RPM; commanded and reported values must be converted from and to rad/sec, respectively. Furthermore, commanded angular velocity must account for the 100:1 gear ratio of the gimbal reduction gear. Eqns. (36) and (37) express the conversion from rad/sec to RPM; conversion from RPM to rad/sec simply requires the measured angular velocity (in RPM) be divided by the conversion factor.

$$\text{Gimbal Rate}\left(\frac{\text{rad}_{gimbal}}{\text{sec}}\right)\times\text{Conversion Factor}=\text{Motor Velocity}\left(\frac{\text{rev}_{motor}}{\text{min}}\right) \tag{36}$$

$$\text{Conversion Factor}=\frac{1\ \text{rev}_{gimbal}}{2\pi\ \text{rad}_{gimbal}}\times\frac{100\ \text{rev}_{motor}}{1\ \text{rev}_{gimbal}}\times\frac{60\ \text{sec}}{1\ \text{min}}=954.93\frac{\text{rev}_{motor}\cdot\text{sec}}{\text{rad}_{gimbal}\cdot\text{min}} \tag{37}$$

The gimbal angles, $\vec{\delta}$, are used to determine the CMG net angular momentum (used to compute the spacecraft dynamics) and the Jacobian matrix (used in the CMG steering law). Both of these calculations require the gimbal angle be expressed in radians; Eqns. (38) and (39) expresses the conversion from gimbal motor encoder count to radians.

$$\delta\left(\text{Encoder Count}\right)\times\text{Conversion Factor}=\delta\left(\text{rad}\right) \tag{38}$$

$$\begin{aligned}\text{Conversion Factor}&=\frac{1\ \text{rev}_{motor}}{4096\ \text{Encoder Counts}}\times\frac{1\ \text{rev}_{gimbal}}{100\ \text{rev}_{motor}}\times\frac{2\pi\ \text{rad}_{gimbal}}{1\ \text{rev}_{gimbal}}\\&=1.534\times10^{-5}\frac{\text{rad}_{gimbal}}{\text{Encoder Count}}\end{aligned} \tag{39}$$

Motor commanding and sampling is performed at 10 Hz using a 100 ms ***Timed Loop***. Figure 35 depicts the *LabVIEW* logic used to transmit and receive motor commands over the CAN bus. Once again, existing EPOS and CANopen libraries simplified the programming process. The top portion of Figure 35 contains the logic for commanding the motors. The gimbal motor commands are the current commanded gimbal rates (as determined by the CMG steering law) scaled by the conversion factor determined in Eqn. (37). The momentum wheel motor commands are simply the user controlled angular velocity from the *CMG Flight Computer.vi* Front Panel (expressed in

RPM). All eight motor velocity commands are passed to *EPOS2 PDOs Write.vi* as an array of 32-bit signed integers.

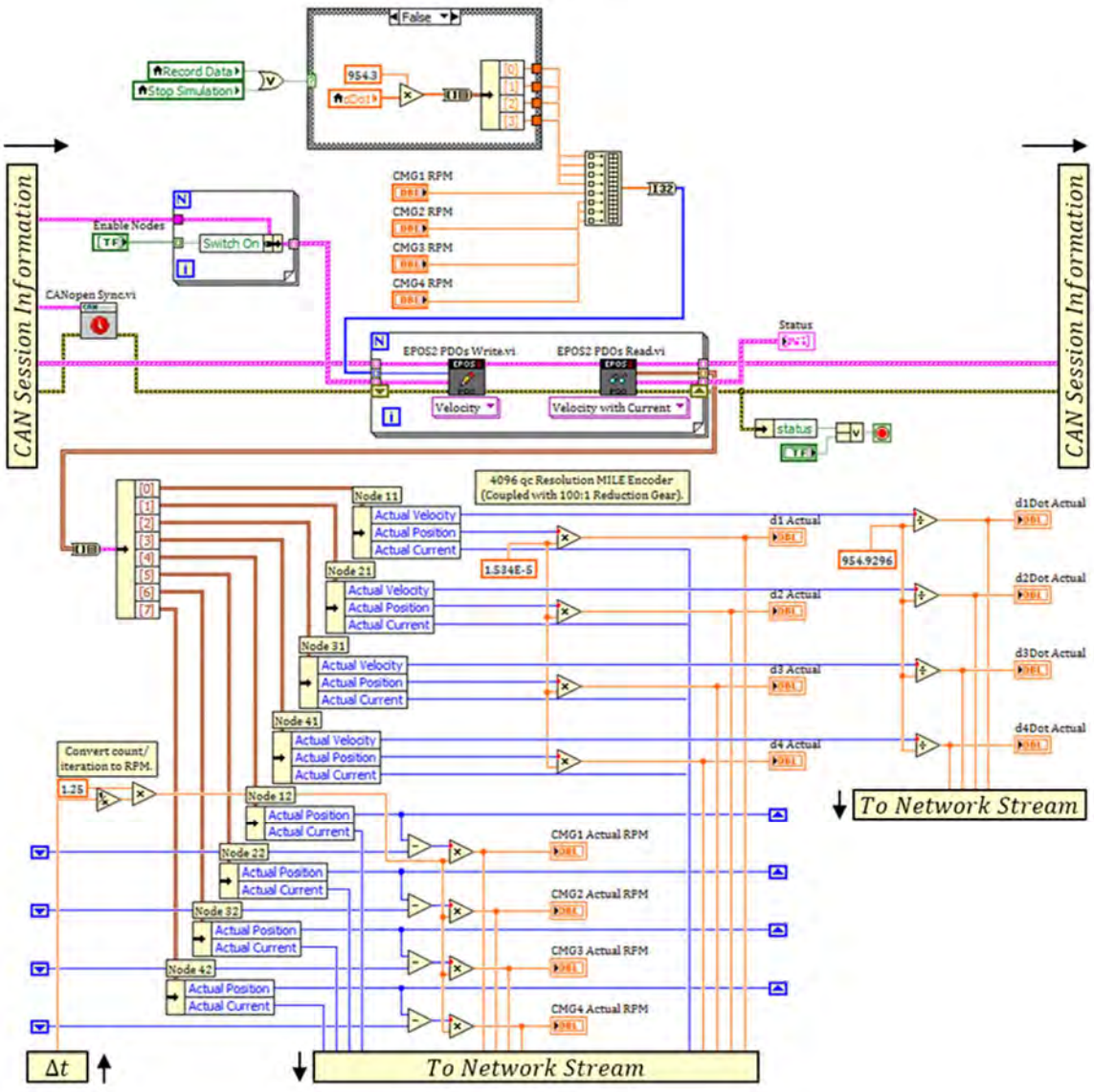

Figure 35. Maxon EPOS2 24/5 motor controller commanding and telemetry sampling in *CMG Flight Computer.vi*

The bottom portion of Figure 35 contains the logic required to process received motor telemetry. *EPOS2 PDOs Read.vi* also includes a custom mode named "Velocity with Current." This subVI outputs an array of eight ***Bundles***, each of which contains values for motor current, motor position, and motor velocity. The gimbal motor angular

velocity is converted to gimbal rate ($\dot{\delta}$ expressed in rad/sec) by dividing the measured value by the conversion factor determined in Eqn. (37). The gimbal motor position is converted to gimbal angle (δ expressed in rad) by scaling the measured value by the conversion factor determined in Eqn. (39). The momentum wheel angular velocity, Ω_{CMG}, is calculated using the approach described in Eqn. (33). The resulting velocity is expressed in RPM and is converted to rad/sec within the net angular momentum logic block (see Figure 31).

H. DATA ACQUISITION

All relevant data is transferred from the embedded flight computer to the host workstation using the Network Stream functionality native to *LabVIEW*. This approach allows for lossless data acquisition and detailed analysis of system performance outside of the real-time environment[6] with minimum resource utilization on the real-time controller [36]. The following telemetry points are captured at a sample frequency of 10 Hz:

- Simulation Time (t_i)
- Quaternion Feedback Control Loop Real-Time Cycle Duration
- CMG Control Loop (CAN Loop) Real-Time Cycle Duration
- Commanded Quaternion ($\bar{q}_c(t_i)$)
- Current Quaternion ($\bar{q}(t_i)$)
- Spacecraft Angular Velocity ($\bar{\omega}_{BN}(t_i)$)
- Commanded Torque ($\bar{u}(t_i)$)
- Commanded Gimbal Rate ($\dot{\delta}_{Commanded}(t_i)$)
- Actual Gimbal Rate ($\dot{\delta}_{CMG}(t_i)$)

[6] All data is saved to a *LabVIEW* .LVM file. This file can be imported into *MATLAB* for data manipulation and plotting. See Appendix B for a sample *MATLAB* import and plotting script.

- Actual Gimbal Position ($\delta_{CMG}(t_i)$)
- Gimbal Motor Current ($I_{Gimbal}(t_i)$)
- Momentum Wheel Angular Velocity ($\Omega_{CMG}(t_i)$)
- Momentum Wheel Motor Current ($I_{Wheel}(t_i)$)
- Singularity Measure ($M(t_i)$)

Figure 36 provides an overview of the data acquisition block that uses Write Elements to Stream logic running within the 10 Hz CAN loop of *CMG Flight Computer.vi*. The Network Stream is currently configured with a first in, first out (FIFO) buffer of 500 elements. While the first column of the .LVM file is a time stamp indicating when the data was written, the inclusion of the simulation time (t_i) data point ensures that data can be analyzed with respect to when it was sampled (vice when it was recorded). The dual time steps are also useful for debugging real-time execution issues. A Boolean trigger (not pictured) ensures that data is only written to the Network Stream while the "Run Simulation" or "Record Data" Boolean controls are set to TRUE (see Figure 25).

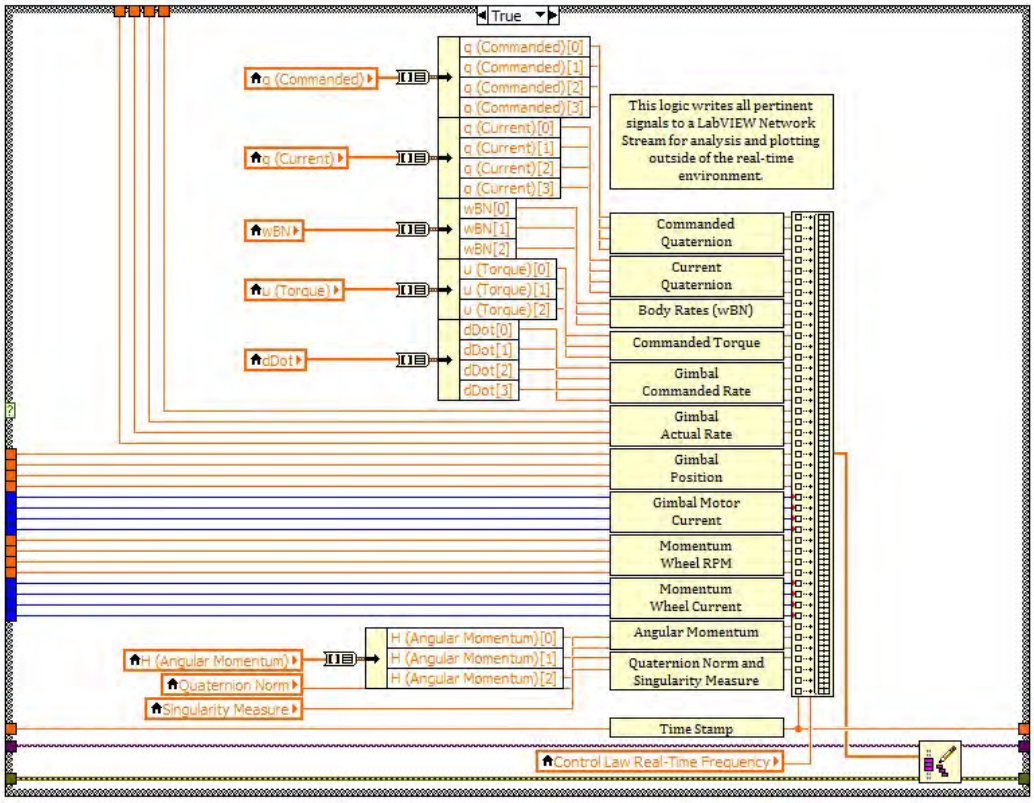

Figure 36. Data acquisition logic in *CMG Flight Computer.vi*

Figure 37 provides an overview of the Read Elements from Stream logic running on the host workstation. *Reader VI.vi* includes logic that first determines if there are 43 elements (a full data set) available on the Network Stream. Once 43 elements are available, a Boolean trigger signals the Write to Measurement File function to execute. If the Read Elements from Stream node does not detect 43 elements within a 10 ms timeout window, the entire loop iterates (without writing any additional data to the .LVM file). The inclusion of the Flush Stream function ensures that all remaining data in the Network Stream buffer is written to the .LVM file before terminating the VI.

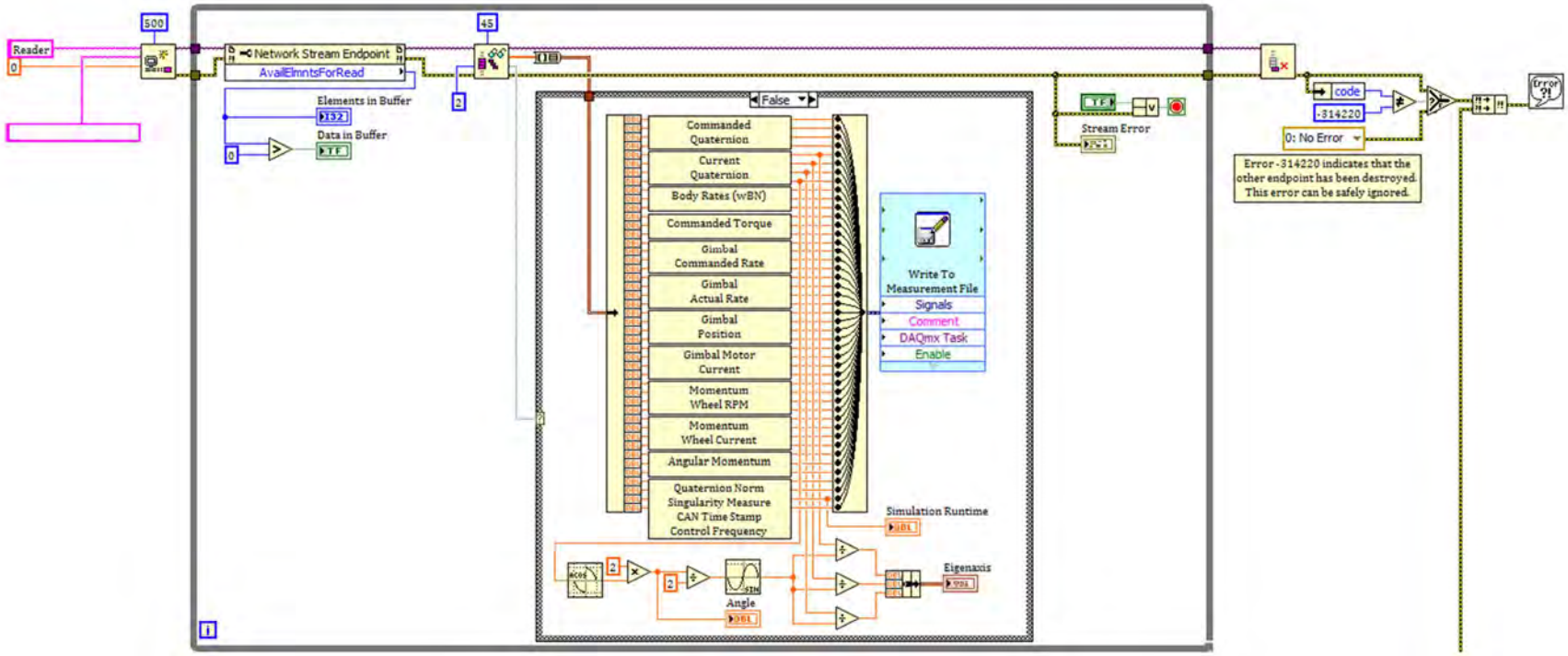

Figure 37. Data acquisition logic in *Reader VI.vi*

The Create Network Stream Reader Endpoint logic in *Reader VI.vi* references a Network Stream endpoint by name and network location (the static IP address of the real-time controller). Each time the *CMG Flight Computer.vi* and *Reader VI.vi* are run, new Network Stream Reader and Writer endpoints are generated. While the flight computer control software is written so that the attitude dynamics and control logic will not iterate until the Network Stream is established (ensuring lossless data capture), the Network Stream logic is only looking for an endpoint with the name "Writer" and the IP address of the real-time controller. This approach allows for multiple instances of *CMG Flight Computer.vi* to be created (each with unique filenames and mass and control system properties representing specific spacecraft and hardware configurations) without the need to create additional *Reader VI.vi* files.

I. ATTITUDE VISUALIZATION

In addition to serving as a Network Stream endpoint, *Reader VI.vi* also contains logic that allows the simulated spacecraft attitude to be visualized in near-real-time. The VI performs an attitude transformation on a 3D representation of the spacecraft using the current quaternion. The *LabVIEW* function *Set Rotation* requires an axis ($\vec{e}$) and angle of rotation (θ), akin to the Eigenaxis rotation discussed in Chapter II. These values are derived from the current quaternion using the following relationships:

$$\theta = 2\cos^{-1}\left(q_4\right) \tag{40}$$

$$\vec{e} = \begin{bmatrix} \dfrac{q_1}{\sin\left(\theta/2\right)} \\ \dfrac{q_2}{\sin\left(\theta/2\right)} \\ \dfrac{q_3}{\sin\left(\theta/2\right)} \end{bmatrix} \tag{41}$$

Eqns. (40) and (41) are performed in the Read Elements from Stream ***While Loop*** (see Figure 37) and are passed to a separate ***While Loop*** (see Figure 38) using local variables.

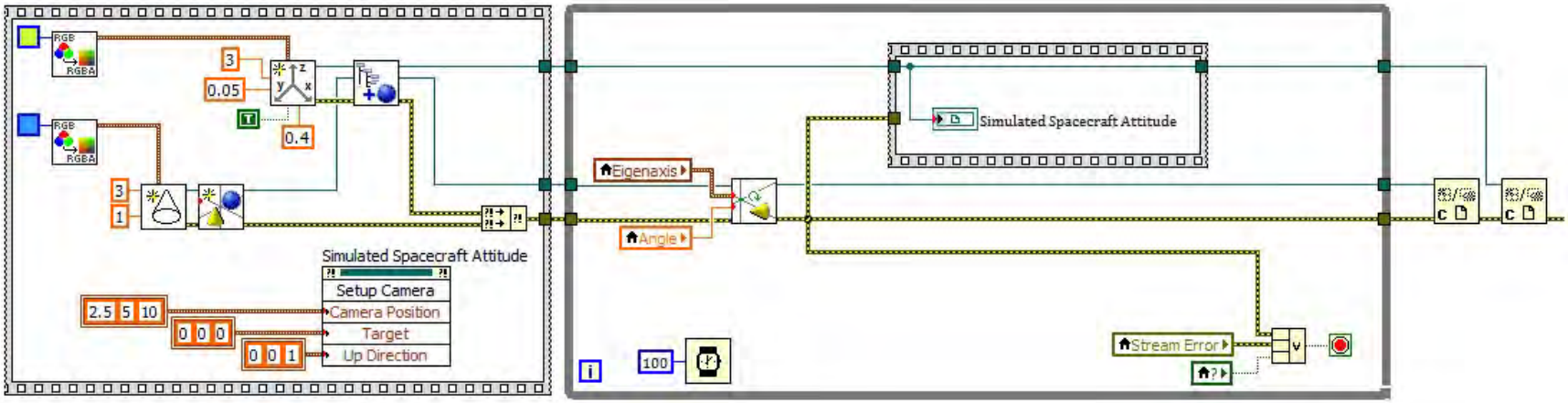

Figure 38. Attitude visualization logic in *Reader VI.vi*

Figure 39 depicts a near-real-time attitude visualization sequence of a maneuver from $\bar{q}_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ to $\bar{q}_f = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^T$ as presented to the experimenter on the *Reader VI.vi* Front Panel. These figures illustrate the simulated spacecraft attitude at $t = 0$ sec, $t = 15.3$ sec, $t = 24.3$ sec, and $t = 90$ sec, respectively.

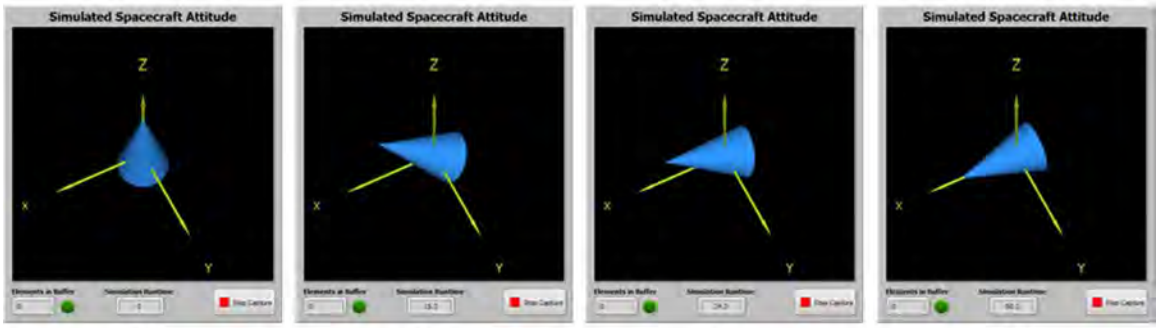

Figure 39. *Reader VI.vi* Front Panel attitude visualization

J. AUTOMATIC GIMBAL POSITION RESET

An additional real-time VI was developed to reset the CMG gimbals to the zero position at the end of each HIL simulation. This VI (see Figure 40) is based around similar logic as the 10 Hz ***Timed Loop*** of *CMG Flight Computer.vi*, but requires no user input to operate. *Reset Gimbal Positions.vi* configures the EPOS2 24/5 motor controllers to operate in position mode instead velocity mode. Once all four gimbal motor controllers are configured, the VI automatically sets the motor controller enable state to ON and commands an encoder position of zero. The Maxon EC 45 Flat gimbal motors include incremental MILE encoders. As such, the encoder home position (otherwise known as the zero position) is set to the position the motor is at when power is applied to motor controller [37]. This home position is maintained regardless of the enable state (ON or OFF) of the motor controllers, allowing the datum to persist between successive iterations of the HIL simulation). All four CMG gimbals should be manually set to the zero position before power was applied to the motor controllers for the first time. *Reset Gimbal Positions.vi* is included in *CMG Flight Compter.vi* as a subVI and is triggered by

pressing the “Reset Gimbals” Boolean control on the *CMG Flight Computer.vi* Front Panel (see Figure 25). This Boolean control stops the 10 Hz ***Timed Loop***, closes the CAN session and FPGA reference established by *CMG Flight Computer.vi*, and executes the subVI. *Reset Gimbal Positions.vi* that sets the enable state of all four gimbal motor controllers to OFF and terminates the subVI once all four gimbal motors return to the home position. The inclusion of *Reset Gimbal Positions.vi* as a subVI is intended to automatically return the CMG gimbals to the zero position at the end of each HIL simulation. The *Reset Gimbal Positions.vi* can be run independently from the *CMG Flight Computer.lvproj* Project Explorer if desired.

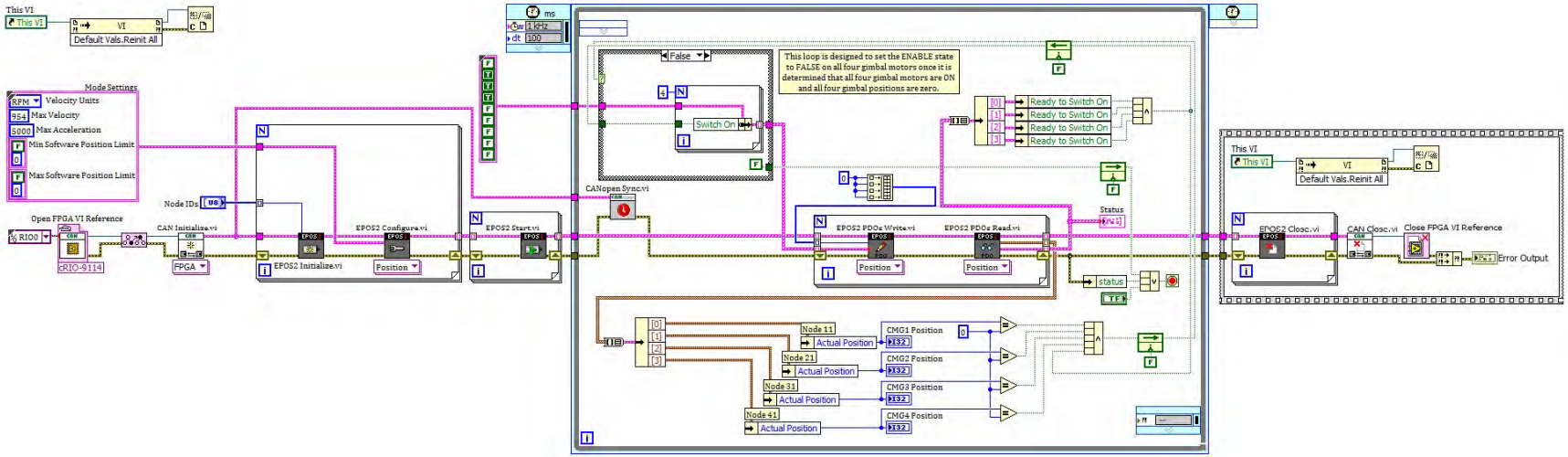

Figure 40. Reset Gimbal Positions.vi Block Diagram

K. SOFTWARE VALIDATION

Initial software-only simulations were performed using the attitude control law and CMG steering law generated in *LabVIEW*[7]. The purpose of these simulations was to verify the performance of the control law and the accuracy of the spacecraft dynamics model using simulated CMG performance. These simulations were performed in real-time and the results saved to a *LabVIEW* Measurement File (.LVM) for further analysis. These results were compared with results obtained using a known-good *Simulink* model.

The control gains for the quaternion feedback control law were set to $k = 1$ and $c = 2.25$. The gimbal rates were limited to the industry-standard $\dot{\delta}_{max} = 1$ rad/sec. Neither the *Simulink* model nor the *LabVIEW* flight software contained a gimbal acceleration limit. Both simulations were performed from an initial attitude of $\bar{q}_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ (spacecraft body axes aligned with the inertial axes) using a commanded quaternion step input of $\bar{q}_c = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^T$. The results of both simulations are included as Figures 41 through 46 and serve to validate the spacecraft dynamics model, quaternion feedback control law, and CMG steering law as implemented in *LabVIEW*. The results from both software-only simulations are nearly identical, as expected. Of note, there is a slight variation in the singularity measure between the *Simulink* and *LabVIEW* models (most apparent at 2.5 seconds in Figure 46). This disparity is assumed to be the result of variations in the way each model propagates the dynamic equations. The *LabVIEW* model uses only Euler steps for integration of the gimbal angles, $\vec{\delta}$, while the *Simulink* model uses a fixed step ode45 solver. This likely results in slight variations in the simulated gimbal angles, which influences the value of the singularity measure.

[7] For the purpose of these simulations, the gimbal positions were simulated via Euler steps using the commanded gimbal rates and the control law time step. This approach assumed infinite acceleration of the gimbal motors. While this assumption does not simulate the actual performance of physical CMGs, it is sufficient to confirm the validity and performance of the real-time attitude dynamics model, quaternion feedback control, and CMG steering law.

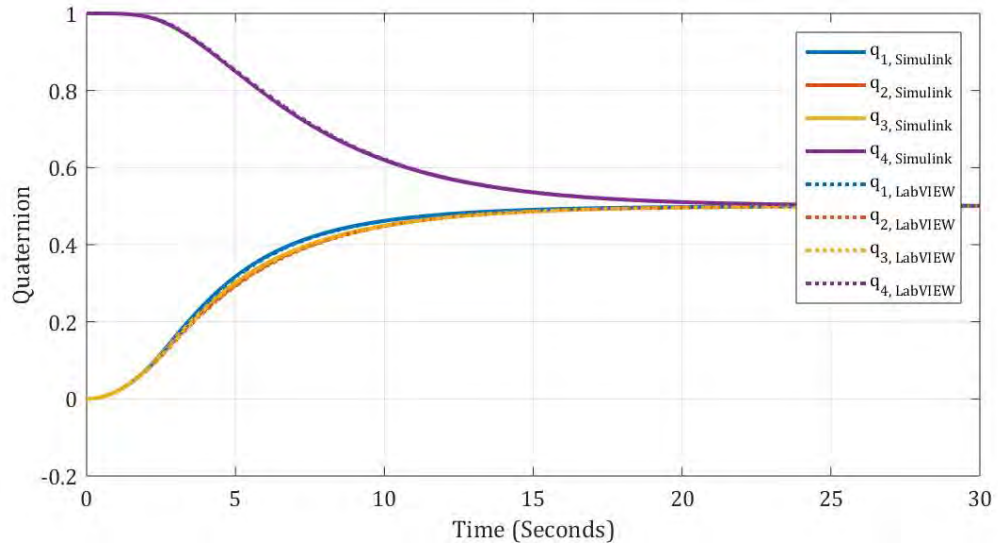

Figure 41. Comparison of simulated quaternion trajectories

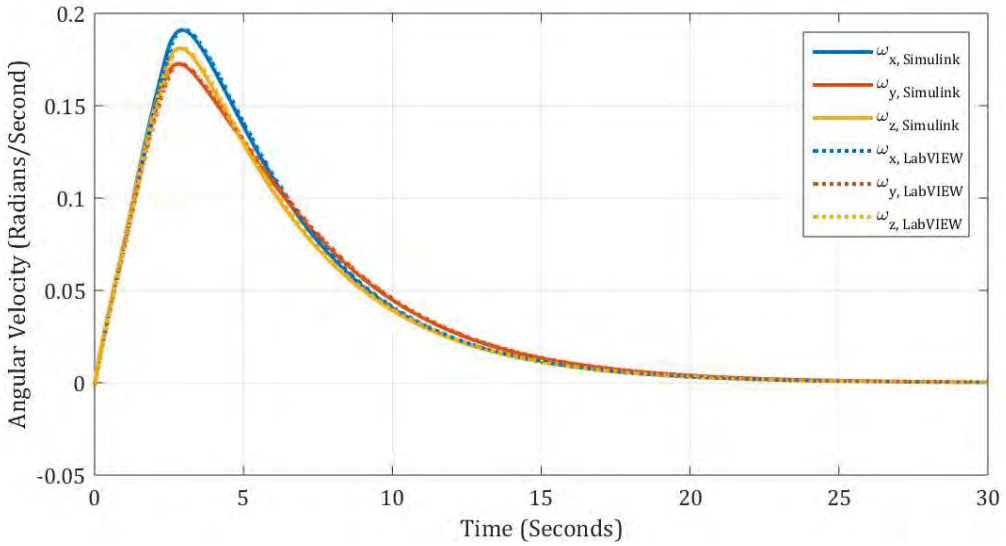

Figure 42. Comparison of simulated angular velocity trajectories

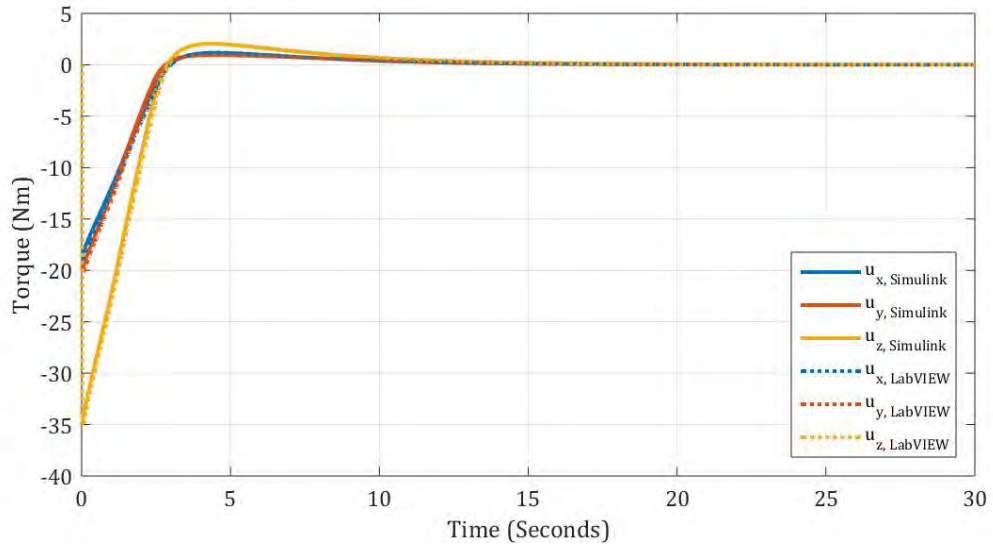

Figure 43. Comparison of torque command signal

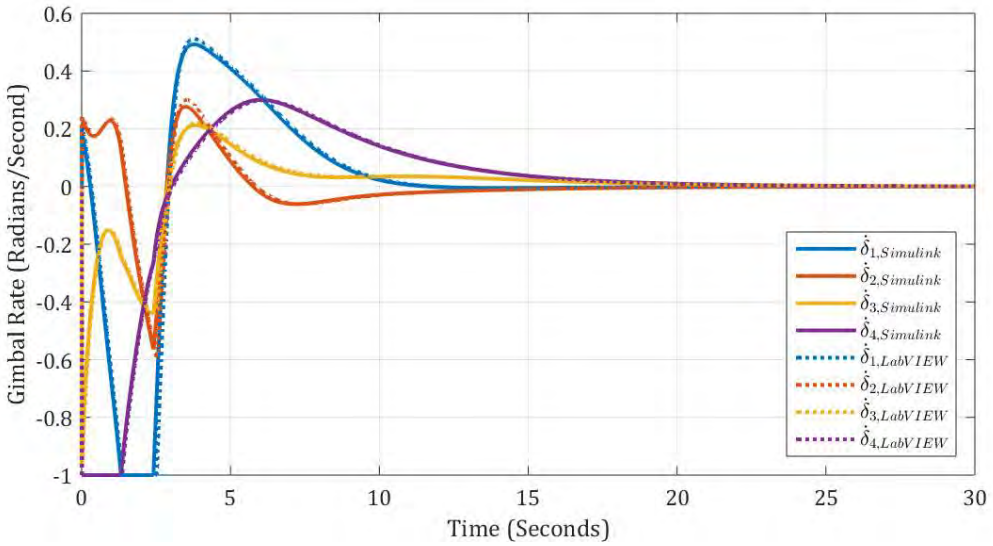

Figure 44. Comparison of simulated gimbal rates

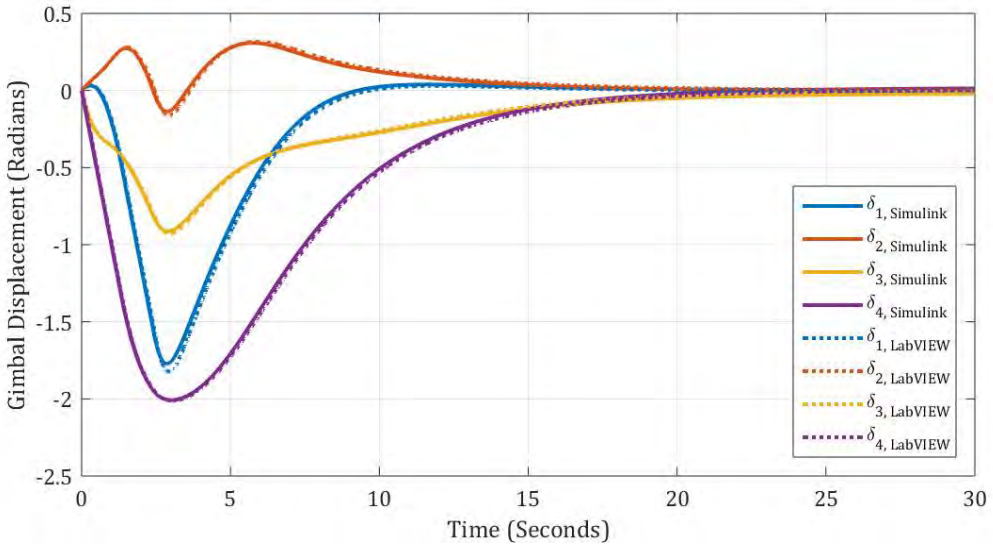

Figure 45. Comparison of simulated gimbal angles

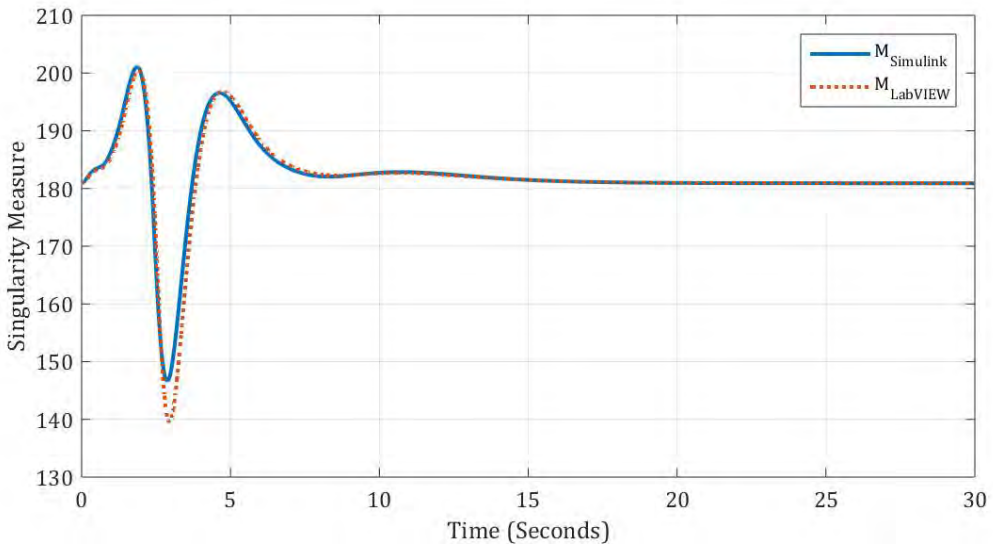

Figure 46. Comparison of CMG singularity measure

L. SUMMARY

The software implementation of an attitude dynamics model, quaternion feedback control law, and pseudoinverse CMG steering law for execution on an NI cRIO real-time controller was described. Various *LabVIEW* Virtual Instruments as well as custom-developed code was used for this task. An overview of the operation and functionality of each developed VI was provided. The results of a software-only validation test performed on the real-time hardware were presented and compared against a known-good *Simulink* model. It was shown that the simulation model implemented on the HIL controller adequately represents the spacecraft dynamics and is suitable for HIL simulation. This paves the way for full HIL attitude control experiments presented in the next chapter.

V. HIL EXPERIMENTS

This chapter presents the results of a series of HIL experiments that were carried out to illustrate the utility of the developed HIL testbed. The results of these experiments, along with data collected from the CMG hardware, are presented and analyzed. First, gimbal motor power consumption at various gimbal rates was evaluated using the previous and updated momentum wheel case design. Next, the performance of the full attitude control system was verified for Eigenaxis maneuvers with CMG skew angles of 54.73° and 90°. The effects of control singularities are also demonstrated. Finally, the impact of the CMG control frequency on control law determinism and gimbal drift is illustrated.

A. GIMBAL MOTOR POWER CONSUMPTION

As discussed in Chapter III, the existing momentum wheel case design was updated to reposition the center of mass so that it lies along the gimbal axis. Final analysis of the updated momentum wheel design in *NX 9* indicated that the center of mass offset was reduced from 19.934 mm to 1.524 mm. In order to evaluate the impact of the momentum wheel case modifications, a series of HIL tests were performed to measure the gimbal motor current required to achieve a desired gimbal rate. Gimbal rates of ± 0.25, ± 0.50, and ± 1.00 rad/sec were commanded using a step input; instantaneous gimbal rate and motor current were recorded at a rate of 10 Hz. Figure 47 compares the performance of the existing and updated momentum wheel designs with respect to positive commanded gimbal rates; Figure 48 provides the same comparison with respect to negative commanded gimbal rates. Table 2 provides the statistical mean (and standard deviation) for gimbal rate and motor current for each test.

The EPOS2 24/5 motor controllers report motor current as a signed value (expressed in mA). The polarity of the current simply indicates the direction of rotation (positive current indicates positive angular velocity) [37]. For ease of comparison (and given the fact that overall power consumption is agnostic of current polarity), all current data is presented as the absolute value of the sampled motor current.

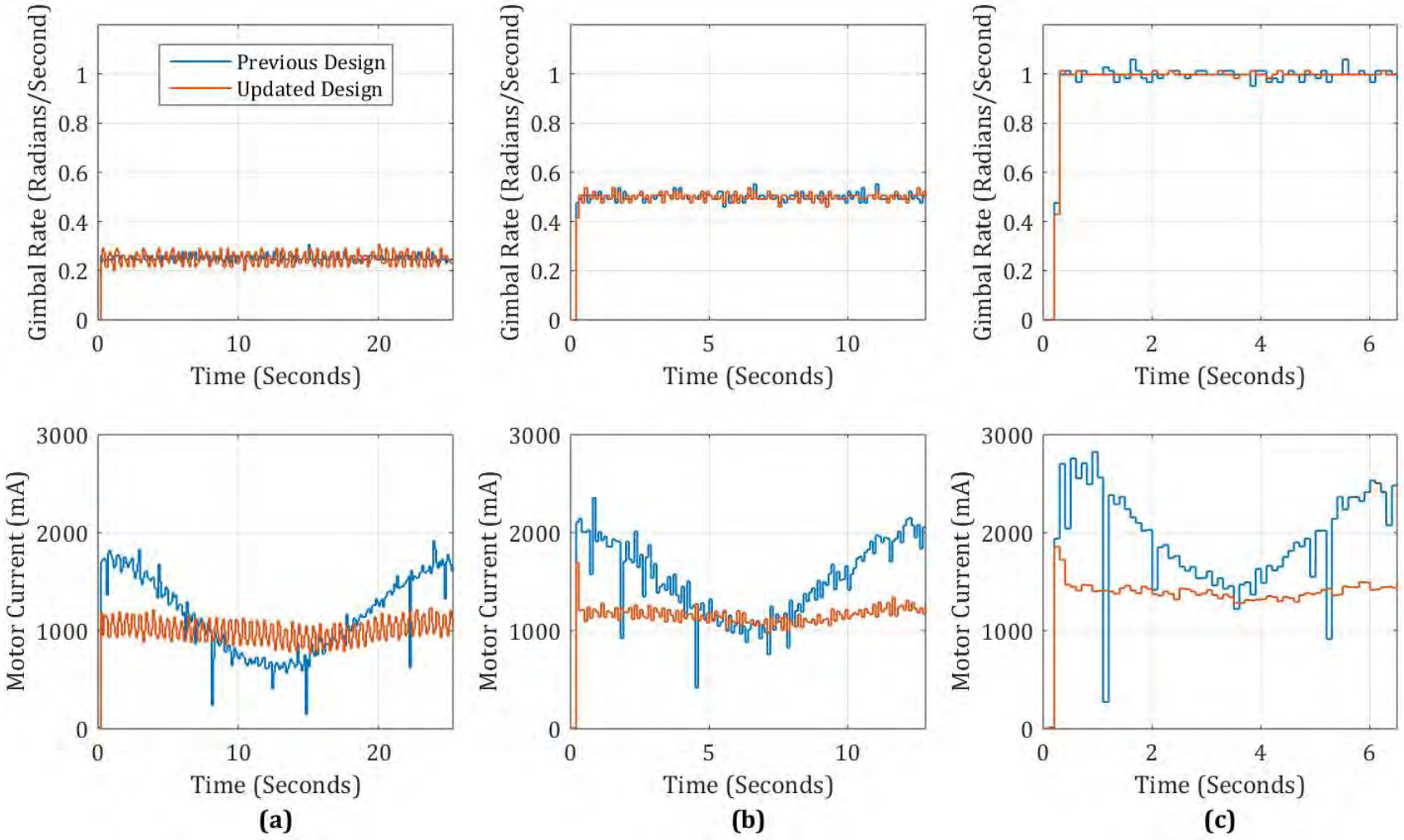

Figure 47. Comparison of gimbal motor current required to maintain positive commanded gimbal rates

(a) $\dot{\delta}_{Cmd}$ = 0.25 rad/sec , (b) $\dot{\delta}_{Cmd}$ = 0.50 rad/sec , (c) $\dot{\delta}_{Cmd}$ = 1.00 rad/sec

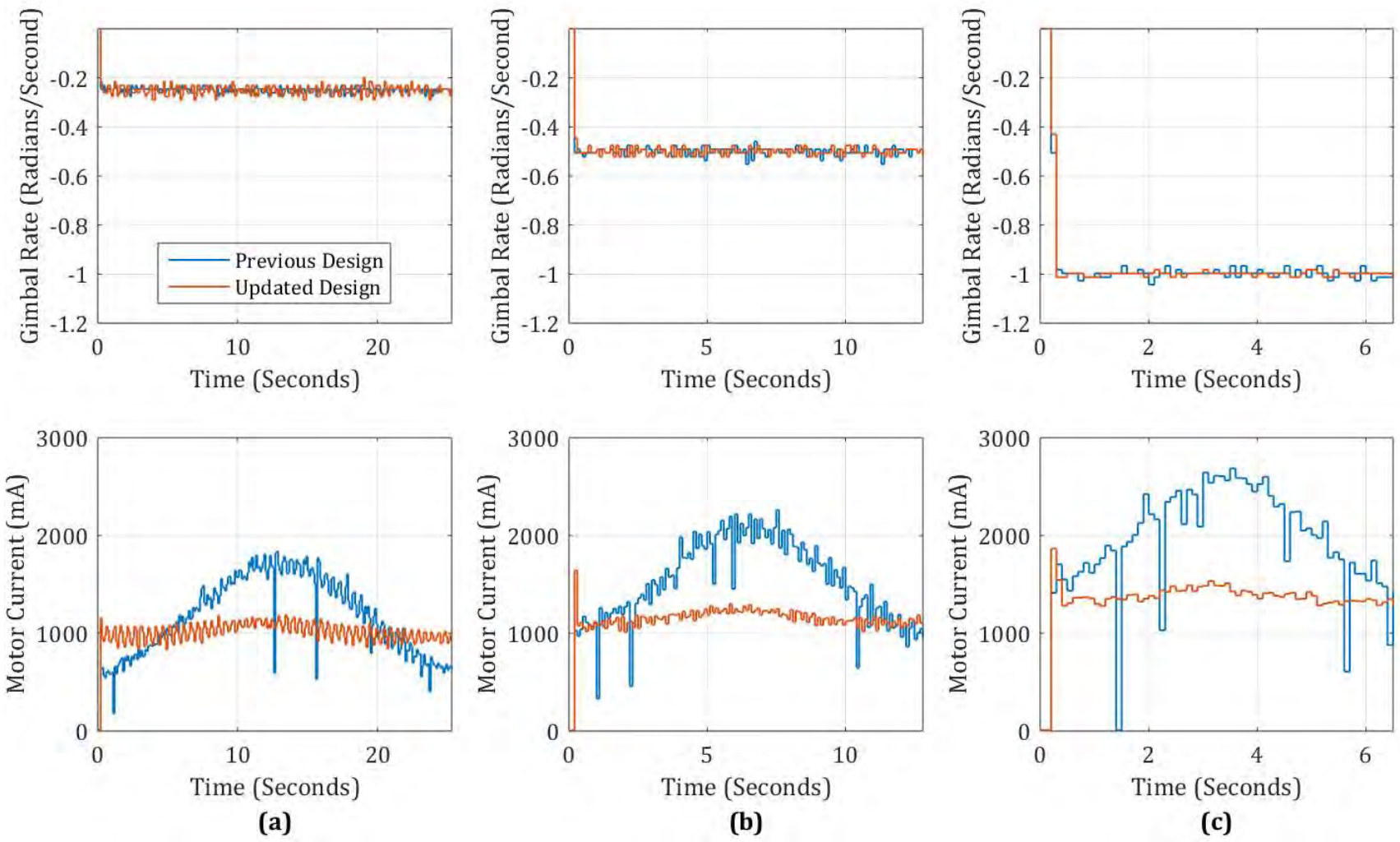

Figure 48. Comparison of gimbal motor current required to maintain negative commanded gimbal rates

(a) $\dot{\delta}_{Cmd} = -0.25$ rad/sec , (b) $\dot{\delta}_{Cmd} = -0.50$ rad/sec , (c) $\dot{\delta}_{Cmd} = -1.00$ rad/sec

Table 2. Summary of gimbal rate and motor current test results

	Previous Design		**Updated Design**	
Commanded Gimbal Rate (rad/sec)	***Mean Gimbal Rate*** (rad/sec)	***Mean Motor Current*** (mA)	***Mean Gimbal Rate*** (rad/sec)	***Mean Motor Current*** (mA)
+ 0.25	0.2503 ± 0.0111	1160.0 ± 394	0.2498 ± 0.0249	1001.9 ± 108
– 0.25	0.2498 ± 0.0107	1177.4 ± 392	0.2507 ± 0.0190	1012.7 ± 83
+ 0.50	0.5000 ± 0.0159	1514.1 ± 386	0.5000 ± 0.0160	1155.8 ± 71
– 0.50	0.5000 ± 0.0157	1537.5 ± 411	0.4999 ± 0.0147	1151.7 ± 66
+ 1.00	0.9976 ± 0.0197	1948.2 ± 488	0.9987 ± 0.0063	1397.8 ± 68
– 1.00	0.9983 ± 0.0167	1975.3 ± 521	0.9998 ± 0.0085	1380.9 ± 66

Visual inspection of Figures 47 and 48 suggests that the updated momentum wheel case design results in more consistent motor current requirements and an overall reduction in power required. These observations are confirmed by the data in Table 2 (which also indicates that system performance is agnostic to gimbal direction). Furthermore, it can be noted that the percentage reduction in required motor current increases with increasing gimbal rate (an approximately 30 percent reduction is realized at ± 1.00 rad/sec as compared with an approximately 15 percent reduction at ± 0.25 rad/sec). While the slight sinusoidal nature of the updated momentum wheel motor current (the orange line in Figures 47 and 48) suggests that there is still a slight imbalance in the design, the rebalancing of the momentum wheel case resulted in a marked improvement in the required current profile. This improvement allows for better characterization of overall system power requirements and a reduction in power consumption fluctuations during attitude control maneuvers. The updated momentum wheel design also allows for tighter control of the commanded gimbal rate, $\dot{\delta}_{Cmd}$, as can be seen by the 68 percent reduction in standard deviation of the measured gimbal rate (when $\dot{\delta}_{Cmd} = 1$ rad/sec).

Figures 47 and 48 also indicate gimbal rate rise time. While the sample frequency of the data obtained limits the fidelity of this characterization, it can be seen that the system achieves the commanded gimbal rate within two sample periods (0.2 sec). While manual tuning of the EPOS2 24/5 internal controller gains could be performed to reduce

gimbal rate rise time, any reduction below 1 sample period (0.1 sec) would be transparent to the control law (since gimbal rate is only measured at discrete points).

B. HIL SIMULATION OF EIGENAXIS SLEW MANEUVERS

A series of attitude control maneuvers was performed to verify the end-to-end functionality of all elements of the HIL testbed. These maneuvers were performed with a momentum wheel angular velocity of 1000 RPM, which resulted in an angular momentum magnitude of $1.55\ \mathrm{N\cdot m\cdot s}$ and a maximum torque capability of $1.55\ \mathrm{N\cdot m}$ per CMG (assuming a gimbal rate limit of 1 rad/sec).

In order to account for this reduced torque capability (which would be characteristic of a smaller spacecraft), the inertia matrix used during software validation was scaled by a factor of 10.

$$J = \begin{bmatrix} 3.725 & 0.059 & 0.005 \\ 0.059 & 3.988 & 0.009 \\ 0.005 & 0.009 & 7.003 \end{bmatrix} \mathrm{kg\cdot m^2}$$

1. Experiment 1: Sample Maneuver with β = 54.73°

The first experiment simulated a rest-to-rest slew maneuver with $\beta = 54.73°$. The control gains for the quaternion feedback control law were set to $k = 2$ and $c = 12.5$. The initial attitude was $\vec{q}_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$ and the commanded quaternion step input was $\vec{q}_c = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}^T$ (the same maneuver performed during software validation – see Chapter IV). The results of this HIL experiment are shown in Figure 49. As can be seen, the responses are similar to those of the software-only simulation, with some small differences as pointed out in the following analysis.

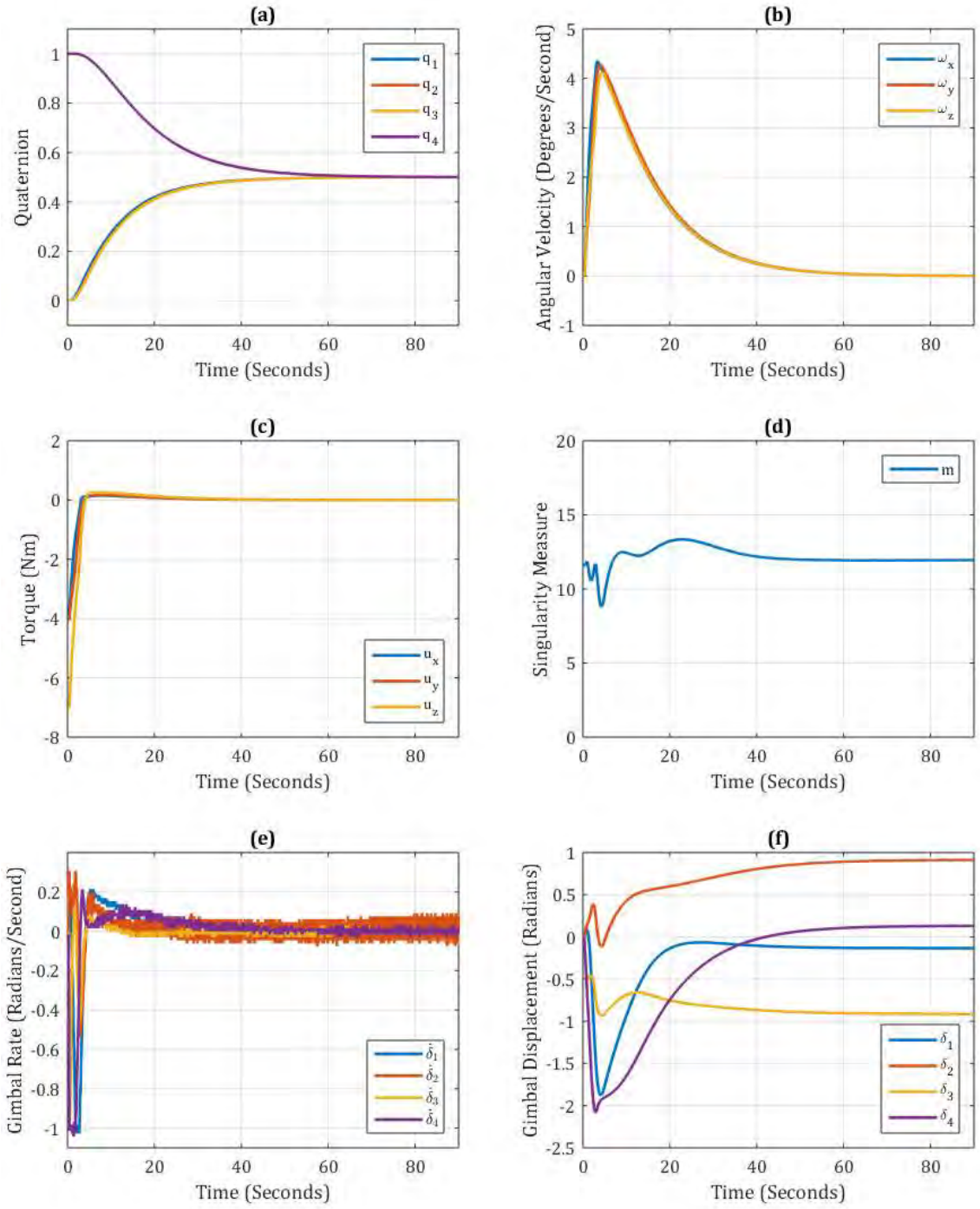

Figure 49. HIL Experiment 1 results ($\beta = 54.73°$, $k = 2$, and $c = 12.5$)

HIL experiment data for: (a) Spacecraft Quaternion, (b) Spacecraft Body Rates, (c) Commanded Torque, (d) Singularity Measure, (e) CMG Gimbal Rates, (f) CMG Gimbal Positions.

The simulated spacecraft quaternion (see Figure 49a) exhibits a critically damped response to the quaternion step input (similar to that of the software-only simulations discussed in Chapter IV). The magnitude of the spacecraft body rates (see Figure 49b) and the commanded torques (see Figure 49c) is reduced as compared with previous software-only simulations due to the reduced angular velocity of the momentum wheels during HIL testing. Magnitude and duration notwithstanding, both the body rates and commanded torques exhibit similar shape profiles to those of software-only testing. The gradual variation of the singularity measure (see Figure 49d) is as expected during gimbal motion. However, the gimbal rate plot (see Figure 49e) shows an unexpected amount of variation with respect to tracking the commanded gimbal rates. Whereas previous software-only simulations resulted in a generally smooth gimbal rate trajectory, the measured gimbal rate contains significant noise. Visual observation of CMG gimbal motion during Experiment 1 did not suggest such behavior, nor does the comparative smoothness of the gimbal trajectories (as shown in Figure 49f). Further analysis suggested that the gimbal rate oscillations observed in Figure 49e are the result of mechanical vibrations induced by imbalance in the momentum wheel rotors. The mechanical vibrations experienced by the CMGs are translated into undesirable rotational motion in the CMG gimbals. The precision of the MILE encoders is such that these small vibrations are being detected as errors in the commanded gimbal rate and the motor controllers are attempting to correct for this error with rapidly changing control inputs (essentially trying to null out the vibration on the gimbal motor shaft). As previously discussed, the EPOS2 24/5 motor controllers contain an internal PI controller operating at a control frequency of 1 kHz, however the motor controllers only report their telemetry over the CAN bus at the CMG control frequency of 10 Hz. As configured, the motor controllers are reporting “Velocity Actual Value” (0x606C in the EPOS Object Dictionary), which is simply the instantaneous motor velocity being fed to the internal PI controller. The EPOS2 24/5 controllers can be configured to report “Velocity Actual Value Averaged” (0x2028 in the EPOS Object Dictionary); however, this value is generated using a first-order digital low pass filter with a cutoff frequency of 5 Hz [37]. The cutoff frequency of the filter cannot be adjusted and therefore limits the utility of this

data (since the cutoff frequency is lower than the CMG control frequency of 10 Hz). The EPOS2 24/5 motor controllers do not allow users to specify a controller dead band (which would allow a specified amount of error in the gimbal rate to be accumulated before applying corrective control input). An additional option would be to reduce the internal controller gains (K_P and K_I), although this would reduce the overall responsiveness of the gimbal motors to gimbal rate commands. The ideal solution requires high-tolerance balancing of the momentum wheel rotors in order to limit the mechanical vibrations experienced by the gimbal motors. It was observed during the manufacturing of the four CMGs that it is difficult to consistently machine and assemble the momentum wheel components (most notably the momentum wheel shaft) to the within the high tolerances required by the prototype design. This resulted in significant variation in the rotational imbalance of the four CMGs and suggests that additional measures be taken to further refine the design for ease of manufacturing in the future.

Fortunately, the gimbal rates are only measured for analysis (they are not currently used by the CMG steering law), and therefore this phenomenon does not impact the overall performance of the attitude control system. For plotting and analysis purposes, an approximate gimbal rate ($\dot{\delta}_{Calculated}$) can be calculated using the difference in reported gimbal position between two successive iterations of the CAN loop. This is the same approach used to calculate the momentum wheel angular velocity. The momentum wheel angular velocity is used to calculate the CMG net angular momentum (used in both the attitude control law and CMG steering law) and therefore must be calculated in real-time. The CMG gimbal rate, however is not used by any control and therefore these calculations can be performed outside of the real-time environment during post-processing of the data. The appropriate computation is:

$$\dot{\delta}_{Calculated}\left(t_i\right) = \frac{d\delta}{dt} = \frac{\delta\left(t_i\right) - \delta\left(t_{i-1}\right)}{t_i - t_{i-1}} \tag{42}$$

Figure 50 compares the sampled gimbal rate ("Velocity Actual Value" reported by the motor controllers) with the calculated gimbal rate (determined using Eqn. (42)). While the gimbal rates appear similar during periods of high acceleration (0 to 4 seconds

in Figure 50), the calculated gimbal rate is much more consistent with the commanded gimbal rate for the remainder of the maneuver. All further gimbal rates plots will utilize the calculated gimbal rate vice the sampled gimbal rate. Visual analysis of Figure 50 indicates that CMG2 experiences the greatest variation in gimbal rate, suggesting that this CMG has the greatest imbalance of the four momentum wheel rotors.

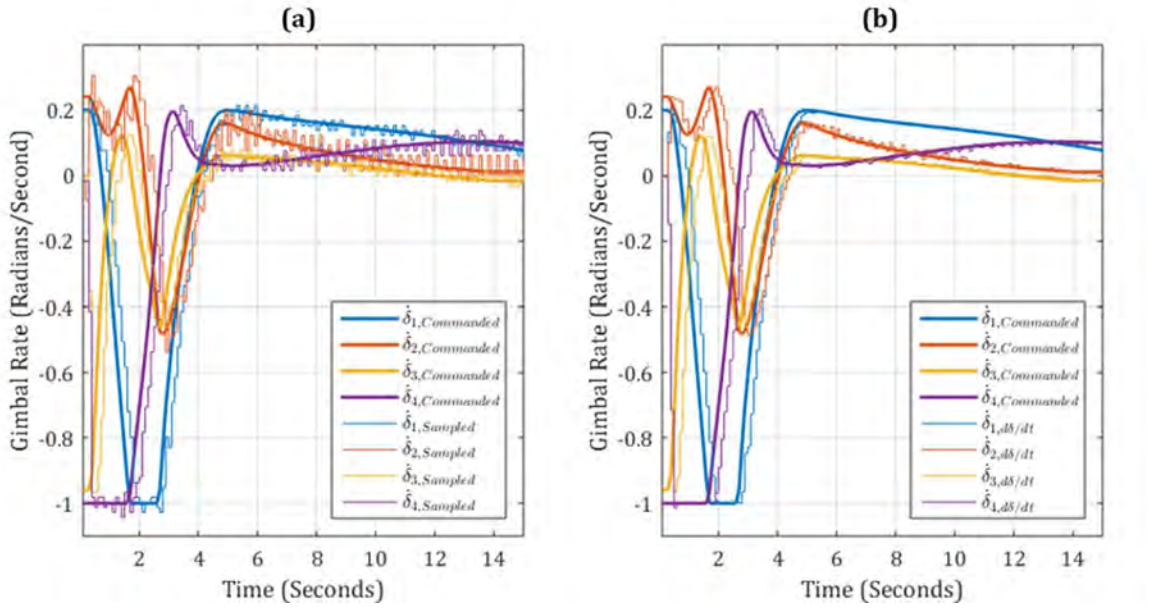

Figure 50. Comparison of sampled and calculated CMG gimbal rate

(a) Sampled gimbal rate ("Velocity Actual Value" reported by EPOS2 24/5 motor controllers); (b) Gimbal rate determined using finite difference of gimbal angles (see Eqn. (42)).

2. Experiment 2: Sample Maneuver with β = 90°

A second HIL experiment was performed using a CMG skew angle of 90°; all remaining simulation parameters were unchanged from Experiment 1. The results of this maneuver are given in Figure 51. Comparison of Figure 50a and Figure 51a suggest that the maneuver requires approximately the same amount of time in either CMG configuration. Minor variations notwithstanding, the spacecraft body rates (see Figure 51b) and the commanded torques (see Figure 51c) follow similar trajectories. This CMG geometry has a control singularity at the origin (as indicated by the singularity measure of 0 at the start of the maneuver in Figure 51d). The initial CMG gimbal motion

breaks the system free of this control singularity and the singularity measure increases (indicating a more controllable state) while the spacecraft body accelerates toward its new attitude. In the absence of any singularity avoidance CMG steering law, the final gimbal positions result in a near-singular control system. Further research could be performed to determine more ideal starting gimbal positions in this CMG geometry. The most noticeable difference between this experiment and Experiment 1 is the gimbal rate and gimbal angle trajectories (see Figure 51e and Figure 51f, respectively). All four CMGs are seen to encounter gimbal rate saturation within the first 3 seconds of the maneuver; this is most likely the result of the control law attempting to steer the CMGs away from the aforementioned singularity at the origin (this period of gimbal rate saturation corresponds with a rapid increase in the singularity measure as seen in Figure 51d).

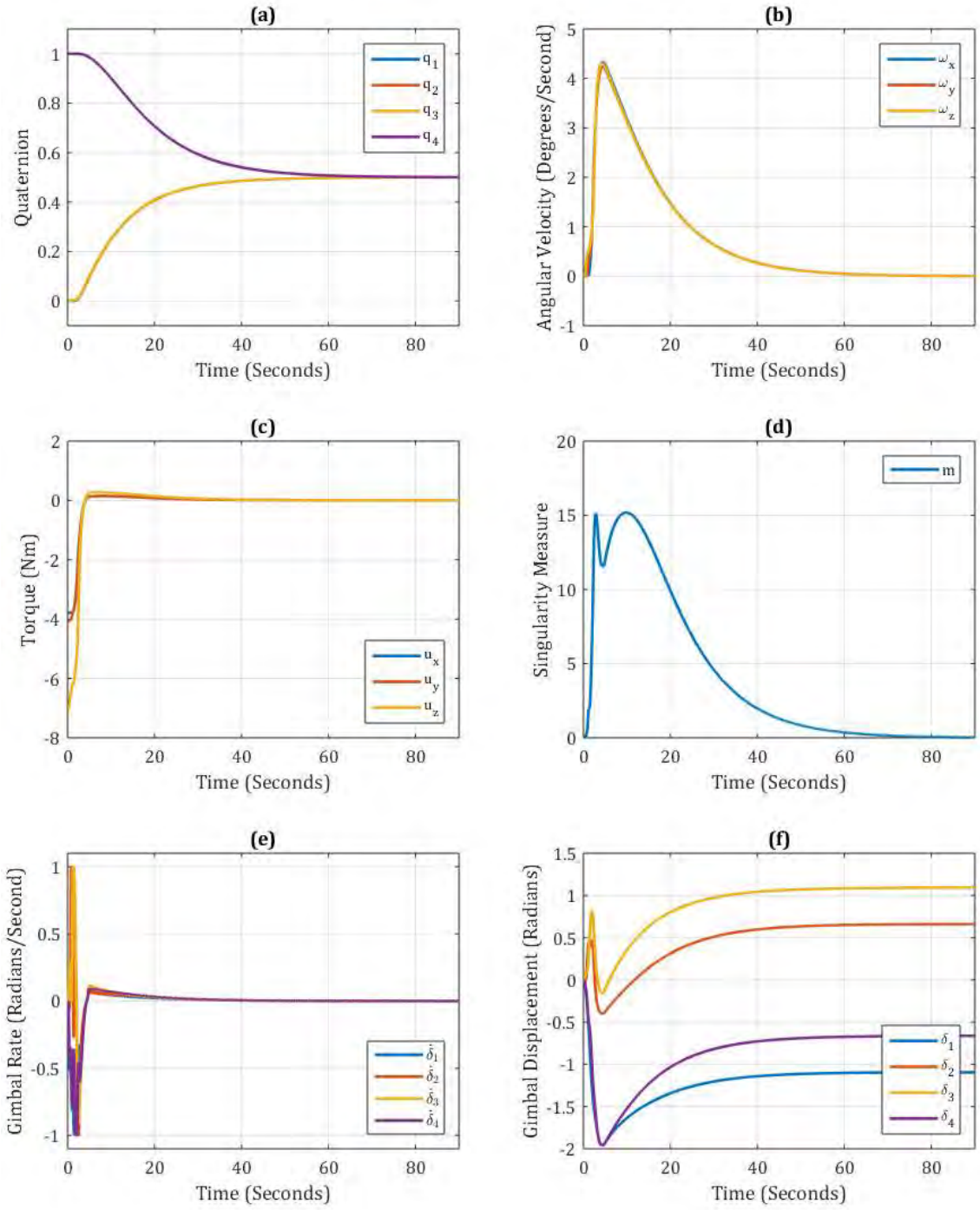

Figure 51. HIL Experiment 2 results ($\beta = 90°$, $k = 2$, and $c = 12.5$)

HIL experiment data for: (a) Spacecraft Quaternion, (b) Spacecraft Body Rates, (c) Commanded Torque, (d) Singularity Measure, (e) CMG Gimbal Rates, (f) CMG Gimbal Positions.

C. IMPACT OF CMG CONTROL FREQUENCY ON GIMBAL REST POSITION

In order for a spacecraft to come to rest at the end of a maneuver, the angular momentum of the spacecraft body, and therefore the net angular momentum of the CMG array, must return to zero[8]. This requirement does not, however, necessitate that the CMG gimbal angles return to zero. In fact, there are an infinite number of gimbal angle combinations which result in a net angular momentum of zero. Yet, since most CMG maneuvers are planned on the ground, knowledge of the initial and final gimbal angles is of absolute importance. Initiating a CMG maneuver from gimbal angles other than those used in planning can result in the spacecraft encountering unexpected control singularities, and this behavior can compromise the mission.

During initial software-only simulations, all four CMG gimbal angles returned nearly to zero (see Figure 45). When the same maneuver was performed using the HIL testbed, all four CMG gimbal angles drifted to non-zero rest values (see Figure 49f). This disparity between software and hardware-based simulations, coupled with the requirement to reliably and consistently predict gimbal rest positions, underscore the value of HIL testing. It is assumed that the finite acceleration of the gimbal motors, as well as the delay between commanding the motors and sampling their response (a function of the control frequency), are the root cause of this gimbal drift.

The flight computer control software was designed with a CMG control frequency of 10 Hz (implemented using a 100 ms cycle period for the CAN bus ***Timed Loop***). The reading and writing of the CAN PDOs (which contain motor commands and motor telemetry) is accomplished using a ***While Loop*** within the ***Timed Loop*** structure. As a result, the ***Timed Loop*** only iterates once the ***While Loop*** is complete and the necessary PDOs have been transmitted to and received from all eight motor controllers. In the event all logic within the CAN bus ***Timed Loop*** is complete prior to the specified cycle period, then the system simply waits to begin the next cycle. However, if the logic within the CAN bus ***Timed Loop*** requires more than the specified cycle period to complete, the next

[8] This assumes zero bias, no external disturbance torques, and no momentum desaturation.

cycle begins late (and the deterministic behavior of the real-time control system begins to deteriorate towards soft real-time). The actual CAN bus cycle duration, which is used to calculate the momentum wheel angular velocity used by the control law (see Eqn. (33)), can also be used to evaluate the real-time CMG control frequency.

$$f_{\text{CMG Control}}(t_i) = \frac{1}{T(t_i)} = \frac{1}{t_i - t_{i-1}} \tag{43}$$

The real-time CMG control frequency for Experiment 1 was calculated using Eqn. (43) and is shown in Figure 52. The statistical mean of the real-time CMG control frequency (when using a CAN loop duration of 100 ms) is precisely 10.000 Hz (standard deviation is 4.3038×10^{-13} Hz). These results confirm the deterministic behavior of the flight computer software as designed.

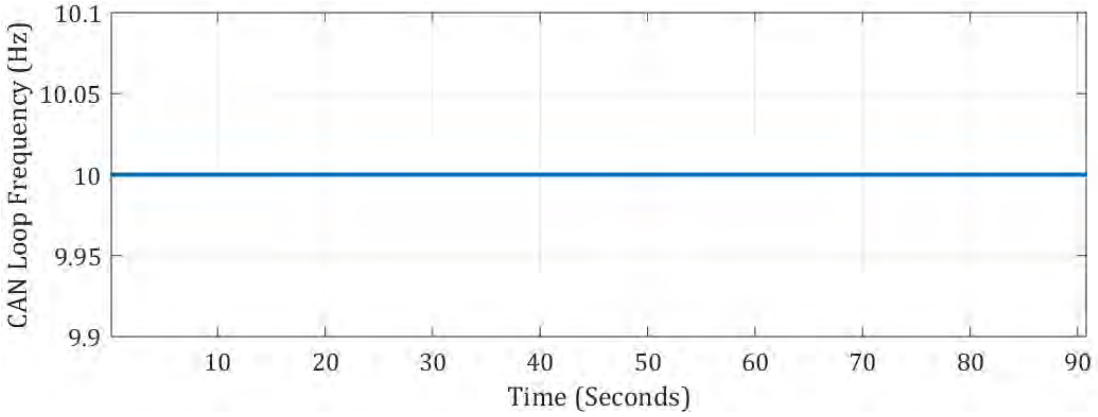

Figure 52. Real-time CMG control frequency with 100 ms CAN loop cycle duration

In the interest of further exploring the effect of sample time on the gimbal rest positions, the CMG control frequency was increased to 100 Hz (by decreasing the commanded CAN bus ***Timed Loop*** cycle period to 10 ms). Experiment 1 was then repeated using this increased CMG control frequency. Figure 53 presents the real-time CMG control frequency (when using a CAN loop duration of 10 ms). Despite the commanded CAN loop duration of 10, the minimum actual loop duration was 16 ms, resulting in a maximum CMG control frequency of about 62.5 Hz. The statistical mean of the real-time CMG control frequency (when using a CAN loop duration of 10 ms) is

48.0454 Hz with a standard deviation of 5.3244 Hz. Thus, when attempting to command the CMGs at 100 Hz, it was only possible to achieve a soft real-time rate of 48 Hz ± 15 Hz 3σ.

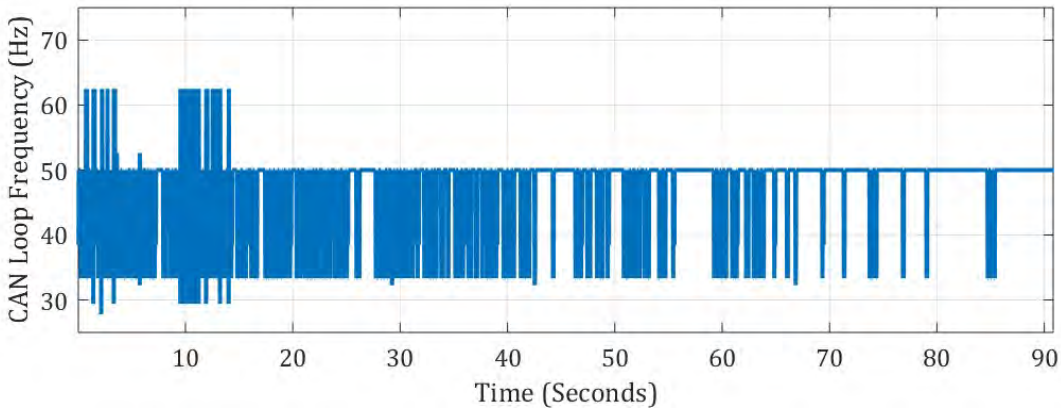

Figure 53. Real-time CMG control frequency comparison

Despite the loss of deterministic performance, the increased CMG control frequency resulted in noticeably different CMG gimbal trajectories than those achieved with a CMG control frequency of 10 Hz. Figure 54 compares the gimbal rest position of Experiment 1 (using both a 10 Hz and 100 Hz CMG control frequency) as well as the results of a software-only simulation.

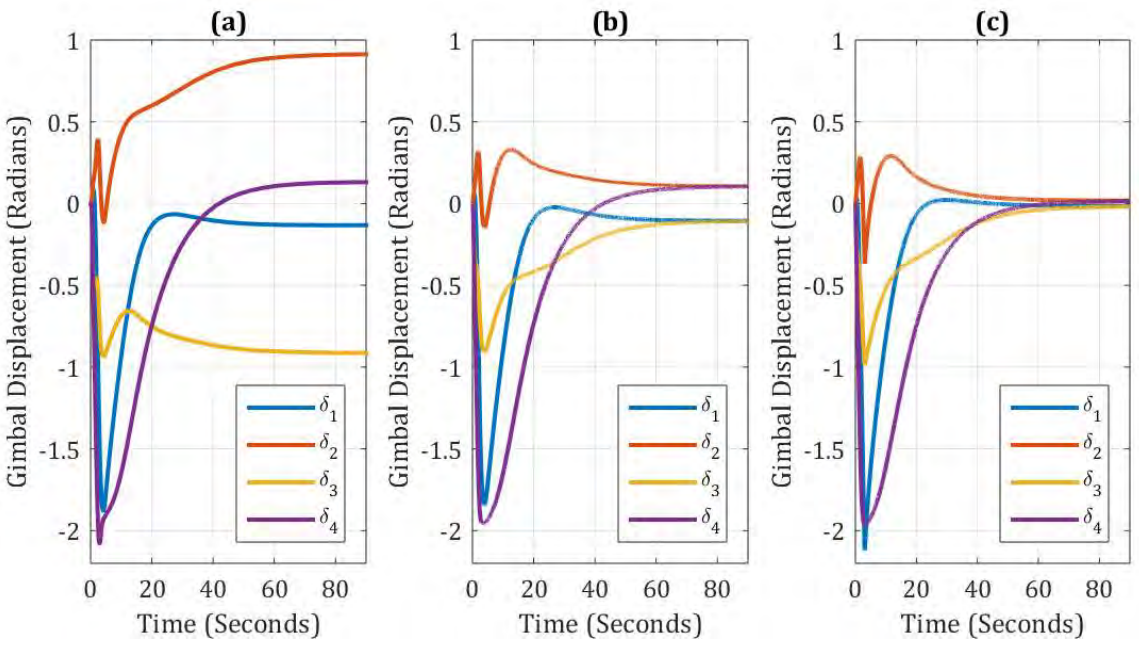

Figure 54. CMG gimbal drift side-by-side comparison for $\beta = 54.73°$

(a) CMG gimbal angle trajectory with CMG control frequency of 10 Hz; (b) CMG gimbal angle trajectory with CMG control frequency of 100 Hz; (c) CMG gimbal angle trajectory from software-only simulation.

While the final gimbal displacement of CMG1 and CMG4 remains relatively unchanged between the two HIL experiments, the final gimbal displacement for CMG2 and CMG3 are significantly different. Neither HIL experiment achieves the near-zero gimbal rest position of the software-only simulation. This suggests that CMG control frequency plays a significant role in the behavior of the CMG steering law. This aspect must therefore be considered as part of the steering law design.

Figure 55 shows the same HIL experimental results as Figure 54a and Figure 54b, but overlays the gimbal trajectories for easier comparison with respect to time. The disparity between the two HIL experiments begins CMG4 at approximately 2.5 seconds into the maneuver, at which point it appears the commanded gimbal rate is decelerating at a rate higher than the CMG hardware can match with a 10 Hz CMG control frequency. The variation in gimbal rates (and therefore gimbal displacement) between the 10 Hz and 100 Hz HIL simulations results in a pseudoinverse control solution that drives the final CMG gimbal displacement to an unexpected configuration. This unexpected variation in

the CMG gimbal rest position can be described as gimbal drift, a phenomenon that is seldom reported in the literature or textbooks.

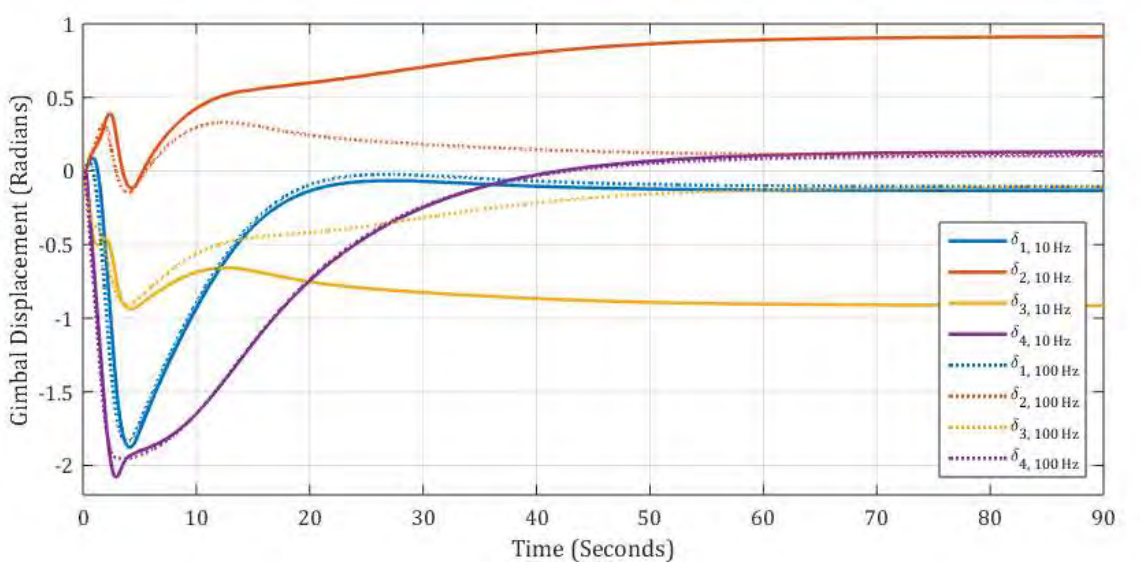

Figure 55. CMG gimbal drift overlay comparison for $\beta = 54.73°$

D. HIL SIMULATION OF MANEUVERS ENCOUNTERING CMG CONTROL SINGULARITIES

In order to evaluate the hardware response to CMG control singularities, the quaternion feedback control gains (k and c) were modified to 1 and 2.25, respectively. Increasing the gains elicited a more aggressive response from the attitude control system and increased the tendency of the CMGs to be driven towards the external singularity surface (defined by the momentum envelope of the array).

1. Experiment 3: Sample Maneuver with β = 54.73°

Unlike Experiments 1 and 2, the simulated spacecraft quaternion (see Figure 56a) exhibits an underdamped, nonlinear response. This overall change in the shape of the response is mostly the result of the modified system gains (rather than due to the control singularity), although the lack of control authority (visible from 4 to 13 seconds in Figure 56b) also results in the spacecraft body gaining excessive angular momentum (thus angular velocity) and overshooting the desired quaternion. While the magnitude of

the commanded torques (see Figure 56c) is smaller than those of Experiment 1 (where $k = 1$, $c = 2.25$, and $\beta = 54.73°$), the quaternion feedback control system is commanding these torques for a significantly longer period (approximately 30 seconds vice 5 seconds in the original maneuver). The application of torque over an extended period results in a greater accumulation of angular momentum in the system (until reaching the external singularity surface). Figure 56d provides a plot of the calculated singularity measure and shows the control system rapidly entering a singular state at approximately 4 seconds and then experiencing a lack of control authority for approximately 9 seconds. Upon recovery, the system completes the attitude maneuver as expected. The control singularity results in erratic, saturated gimbal rates (see Figure 56e) and a series of rapid oscillations in the gimbal angle (see Figure 56f) which can reduce the life of the hardware. In addition to the loss of attitude control authority, singularities result in non-deterministic behavior of the gimbals results and unpredictable gimbal rest positions (compare Figure 56f to Figure 49f). As previously discussed, uncertainty in the gimbal rest positions following a maneuver complicates the planning of CMG maneuvers from the ground and can increase the risk of future control singularities.

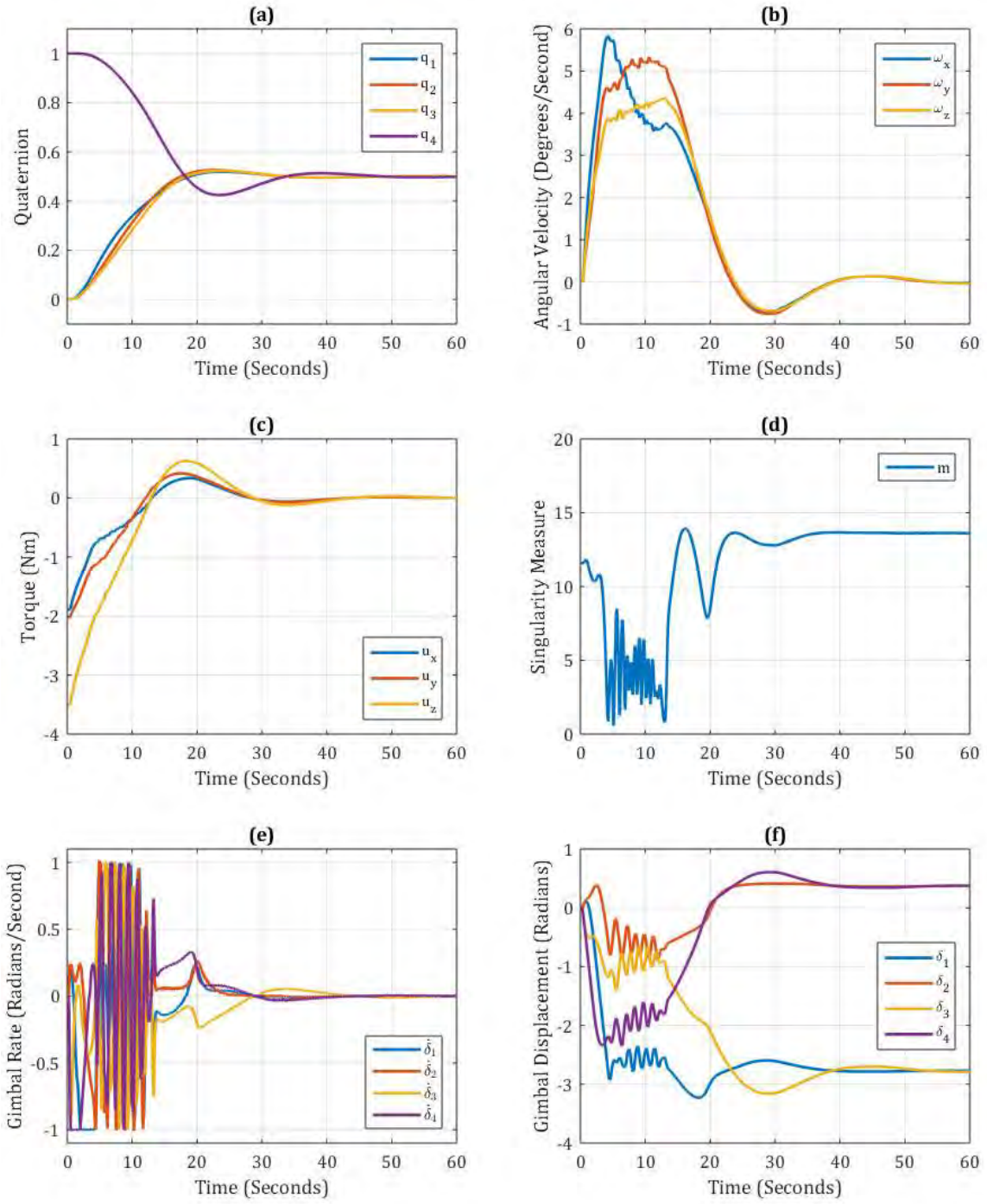

Figure 56. HIL Experiment 3 results ($\beta = 54.73°$, $k = 1$, and $c = 2.25$)

HIL experiment data for: (a) Spacecraft Quaternion, (b) Spacecraft Body Rates, (c) Commanded Torque, (d) Singularity Measure, (e) CMG Gimbal Rates, (f) CMG Gimbal Positions.

2. Experiment 4: Sample Maneuver with β = 90°

Figure 57 shows the results of the last HIL experiment with $k = 1$, $c = 2.25$, and $\beta = 90°$. The simulated spacecraft quaternion (see Figure 57a) exhibits a similar underdamped response as the previous Experiment 3 (where $k = 1$, $c = 2.25$, and $\beta = 54.73°$). The spacecraft body rates (see Figure 57b) are smaller in magnitude than Experiment 3 when the first control singularity occurs, however the loss of control authority is also shorter in duration so that rate buildup is reduced. The commanded torques (see Figure 57c) are once again smaller in magnitude than Experiment 2 (due to the change in the values of the controller gains) and similar in shape profile to Experiment 3. Unlike the previous experiment, the simulation encounters multiple singularities, as indicated by the multitude of near-zero points on Figure 57d. Each singularity once again results in rapid gimbal actuation (see Figure 57e) and this produces unpredictable gimbal trajectories (see Figure 57e). The particular combination of controller gains and CMG skew angle used in this experiment left the attitude control system unable to fully bring the spacecraft rest at the desired attitude.

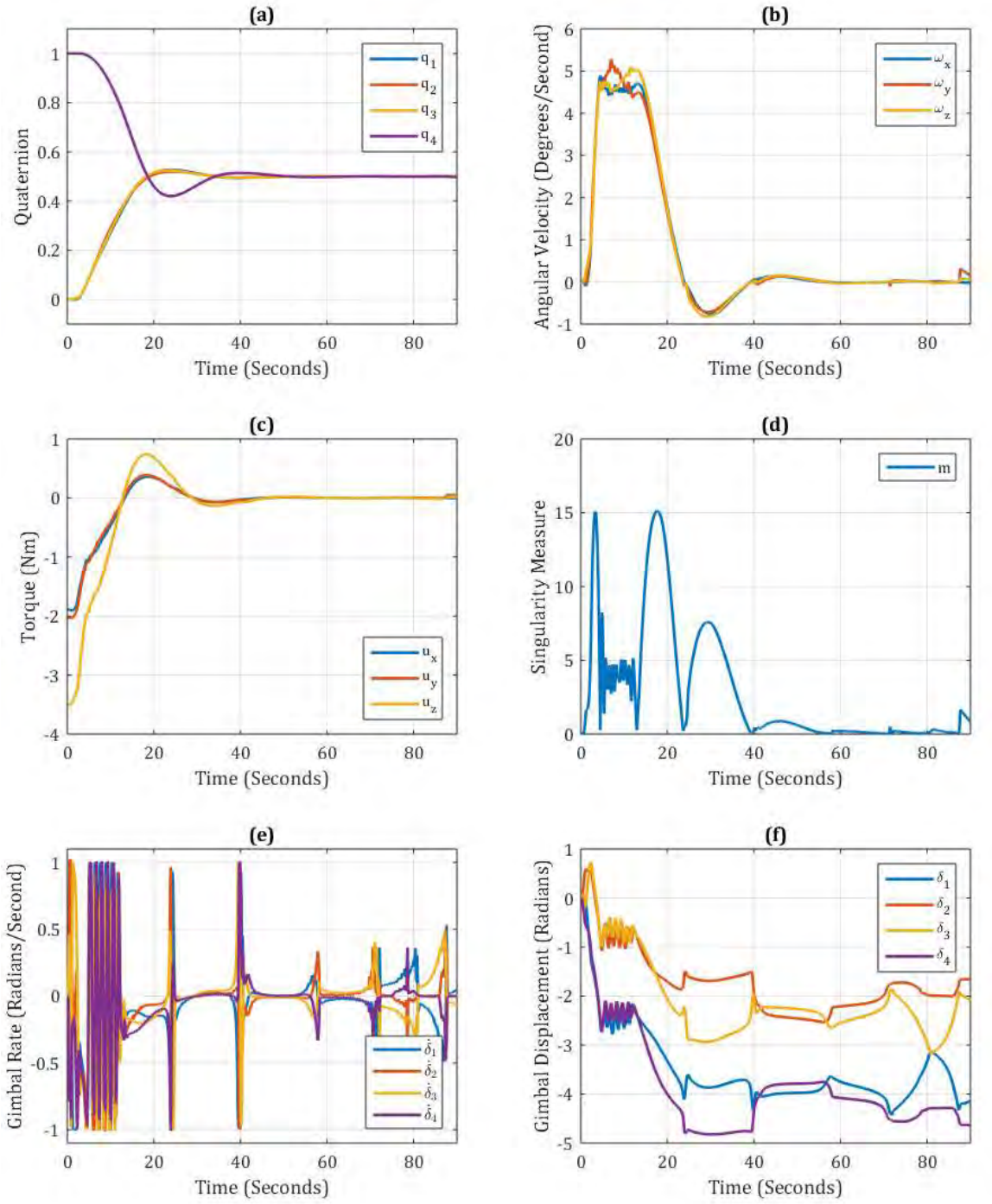

Figure 57. HIL simulation results ($\beta = 90°$, $k = 1$, and $c = 2.25$)

HIL experiment data for: (a) Spacecraft Quaternion, (b) Spacecraft Body Rates, (c) Commanded Torque, (d) Singularity Measure, (e) CMG Gimbal Rates, (f) CMG Gimbal Positions.

The normalized momentum trajectories from HIL Experiments 1 through 4 were plotted within the singularity surfaces for each array (see Figure 58) to illustrate how the momentum vectors interact with the singularity surface. As expected, the net momentum trajectories are approximately parallel with the maneuver axis. Referring to Figure 58, the momentum trajectories of the Experiments 1 and 2 are relatively smooth and remain clear of the external singularity surface. Due to the increase in controller gain, both Experiments 3 and 4 encounter external control singularities which result in off-nominal and unpredictable behavior. The momentum trajectories are observed to skip along on the singularity surface similar to a rock skipping over water.

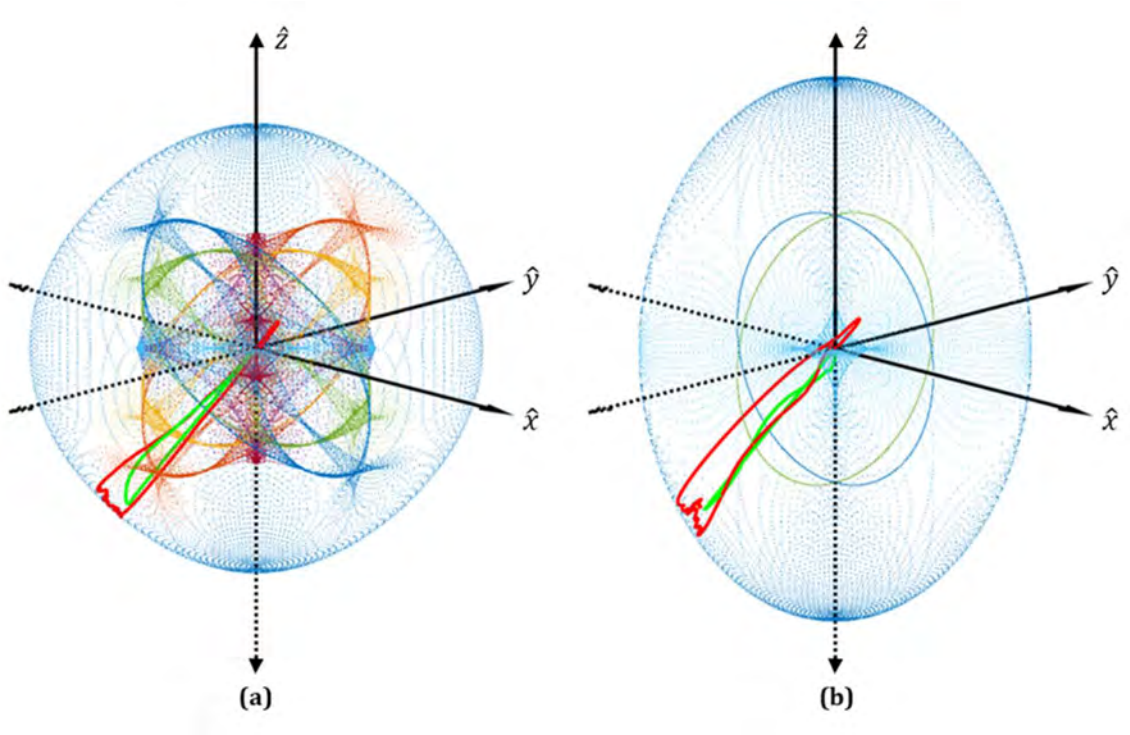

Figure 58. Normalized momentum trajectory within singularity surface

(a) Experiment 1 normalized momentum trajectory (green line) and Experiment 3 normalized momentum trajectory (red line) within singularity surface of an array with $\beta = 54.73°$; (b) Experiment 2 normalized momentum trajectory (green line) and Experiment 4 normalized momentum trajectory (red line) within singularity surface of an array with $\beta = 54.73°$;

E. SUMMARY

The results of several HIL experiments using the developed HIL testbed were presented in this chapter. Initial experiments verified end-to-end system functionality and characterized the impact of the momentum wheel case redesign with respect to system power consumption. Further testing analyzed the real-time performance of the flight computer control software and examined the impact of increased CMG control frequency on real-time performance and hardware response. Finally, a series of tests were performed to capture the overall system response to CMG control singularities.

Overall, the experiments presented in this chapter illustrate the utility of HIL testing, as such tests expose read-world phenomenon that are easy to overlook in modeling and software-only simulations. This testing emphasized that it can be difficult to fully capture the nature of physical components in a simulation environment. In this spirit, HIL testing is vital to help refine the simulation models in order to reduce discrepancies between design studies and practical implementation.

VI. CONCLUSIONS AND FUTURE WORK

CMGs are remarkably capable attitude control actuators but are often discounted during the spacecraft design process due to the relative complexity of their steering law and the possibility of control singularities. Increased use of CMGs in modern spacecraft therefore requires the continued development of singularity-robust steering laws. These new algorithms must be thoroughly tested and vetted using ground-based attitude control simulations. While software-based simulations are a logical starting point, software-only tests can fail to capture the nuanced real-world performance of physical hardware. The goal of this thesis was to help bridge the gap between simulation and practice by creating a reconfigurable CMG array and HIL testbed that can be used by students and other researchers at NPS to rapidly iterate and evaluate attitude control and CMG steering algorithms as well as techniques for optimal attitude guidance, in a hardware setting.

A. SUMMARY OF WORK

An open-architecture attitude control system was developed. Four single-gimbal CMGs were constructed using mechanism designs developed previously. Minor modifications were made to the previous CMG designs based on observations made during prototype testing. These modifications reduced the complexity of the momentum wheel shaft design and resulted in an overall reduction in gimbal motor power consumption through rebalancing the momentum wheel along the gimbal axis. A power and communications architecture was developed to support real-time command and telemetry sampling of eight motor controllers via a CAN bus. A collection of VIs was developed in *LabVIEW* System Design Software in order to implement quaternion feedback control and a simple pseudoinverse CMG steering law on an NI cRIO real-time controller. A spacecraft attitude dynamics model was also included on the embedded flight computer which allowed HIL attitude control experiments to be performed with real CMG hardware. Telemetry is transmitted to a host workstation for further analysis and near-real-time attitude visualization. A series of HIL attitude control experiments was performed to illustrate the utility of the developed HIL testbed. As a part of this

experimentation, a relationship between the CMG control frequency and gimbal angle drift was observed. This phenomenon would go unnoticed if only software simulations of the CMG attitude control system were performed, thus underscoring the need for experimentation with real hardware.

B. FUTURE WORK

The process of machining and integrating the momentum wheel shafts underscored several areas for continued design improvement. The current design utilizes a cadmium-plated steel lock nut to secure the momentum wheel to the momentum wheel shaft. Installation of this lock nut requires a ratchet with a deep socket (to accommodate the momentum wheel shaft). The mechanical advantage of the ratchet, material imperfections, and variation in the torque required to tighten the lock nut all contribute the risk of warping the momentum wheel shaft during installation. This warping could result in undesirable shaft runout with respect to the momentum wheel axis of rotation and would cause unnecessary wear on both the momentum wheel motor and the precision duplex bearing set. Future work should consider a redesign to the momentum wheel shaft to allow for increased precision and repeatability during the machining and assembly process. Consideration should be given to integrating the brushless DC (BLDC) motor magnets onto the momentum wheel shaft and passing the entire shaft through the BLDC coils. This design would require the use of bearings at each end of the momentum wheel shaft (as opposed to the single duplex bearing set used in the present design).

All four momentum wheels were originally balanced to within ISO G-0.4 standards for gyroscopes. Unfortunately, a discrepancy in the required tolerance between the momentum wheel shaft and the inner race of the Timken precision duplex bearing set was discovered during final assembly of the momentum wheel enclosures. As a result, four new momentum wheel shafts were machined but there was insufficient time to rebalance the momentum wheels prior to the initial testing required for system evaluation. In order to minimize the wear on the momentum wheel motors due to rotor imbalance, it is suggested the momentum wheels be operated at a reduced angular velocity. It is recommended that the currently-installed momentum wheels be rebalanced

before being operated at the nominal angular velocity of 5000 RPM. Alternatively, the entire momentum wheel shaft and rotor assembly could be redesigned to reduce the impact of part variance of system performance.

The modular nature of the embedded flight computer control software allows for easy modification or addition of features. Additional logic can be generated to implement alternate attitude control algorithms or singularity-robust CMG steering laws. Furthermore, the spacecraft attitude dynamics model can be improved to include external disturbances or orbital motion during HIL attitude control experiments (neither of which can be modeled using ground based three-axis simulators). Finally, logic can be added to *Reader VI.vi* to generate an inter-process data stream based on the AGI *Systems Tool Kit (STK)* Connect message format. This logic would allow the simulated spacecraft attitude to be passed to an *STK* scenario for improved attitude visualization, especially during simulations of imaging satellite collection maneuvers.

Use of the CMG system with the NPS R-SAT simulator will require the incorporation of real-world attitude sensor data. The existing KVH DSP-3000 fiber optic gyros pass a digital bit stream over a DE-9 serial interface [38] and can be integrated with the cRIO 9024 real-time controller using an NI 9870 4-Port, RS-232 serial interface module. Software logic for gyro signal processing and filtering can then be built using existing functions within *LabVIEW*.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MAXON EPOS2 24/5 MOTOR CONTROLLER CONFIGURATION

This appendix details the required settings for the Setup Wizard tool in Maxon *EPOS Studio*. Although the momentum wheels and CMG gimbals are driven by identical motors, the exact configuration of the associated EPOS2 24/5 motor controllers is a function of both the motor and the position sensor. The variation in motor controller configuration is initiated during Step 4: Motor Commutation Type. The momentum wheel motors controllers (Nodes 12, 22, 32, and 42) utilize *Block* commutation; the gimbal motor controllers (Nodes 11, 21, 31, and 41) utilize *Sinus* commutation. Additionally, as described in Chapter IV, the maximum angular velocity of the gimbal motors is limited to 3125 RPM (vice the hardware limit of 10000 RPM) due to the use of a sinusoidal commutation command signal. Screenshots illustrating the configuration steps are given in Figures 59 through 66 for the gimbal motors and Figures 67 through 73 for the momentum wheel motors. Step 1 of this process simply asks the user to confirm that they are familiar with the operation of the motor and motor controller; no additional user input is required.

(1) Gimbal Motor Controller Configuration

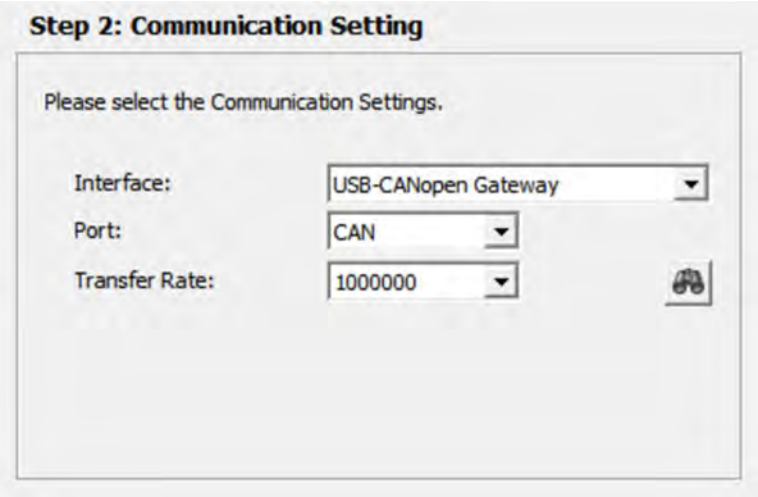

Figure 59. Gimbal motor setup (Step 2)

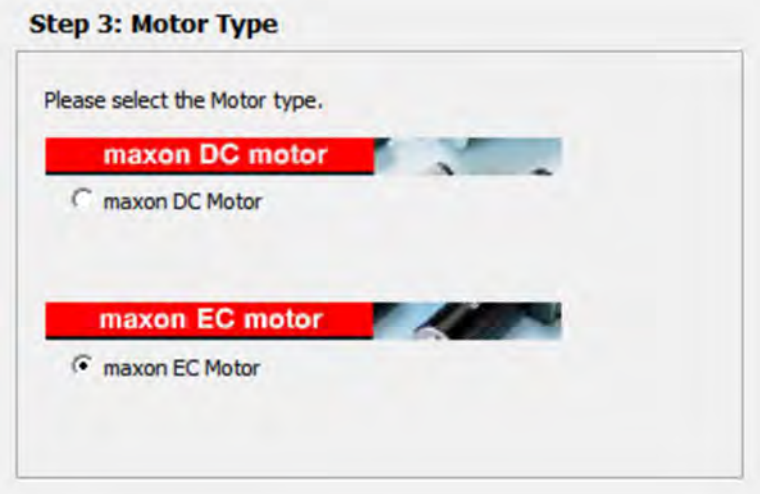

Figure 60. Gimbal motor setup (Step 3)

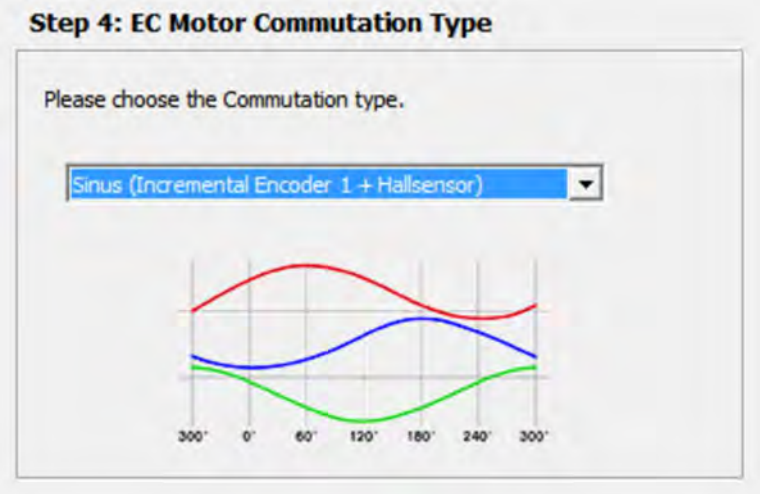

Figure 61. Gimbal motor setup (Step 4)

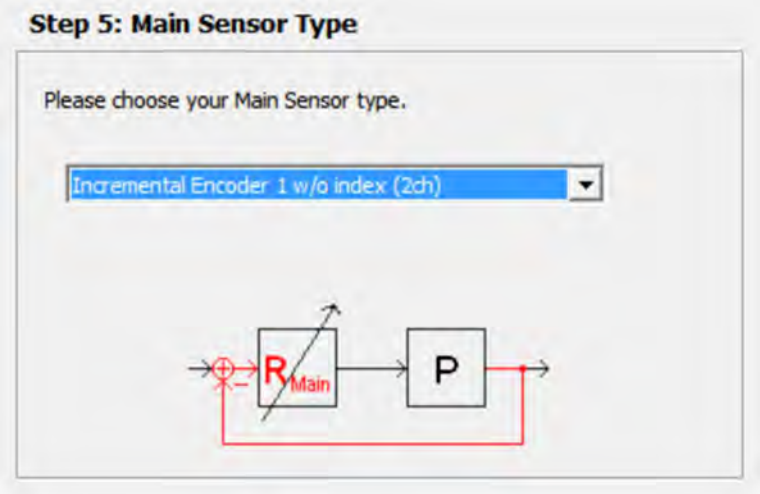

Figure 62. Gimbal motor setup (Step 5)

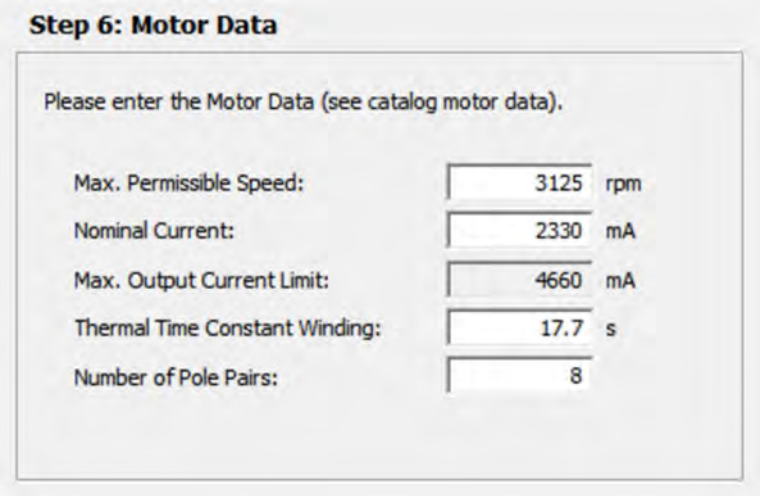

Figure 63. Gimbal motor setup (Step 6)

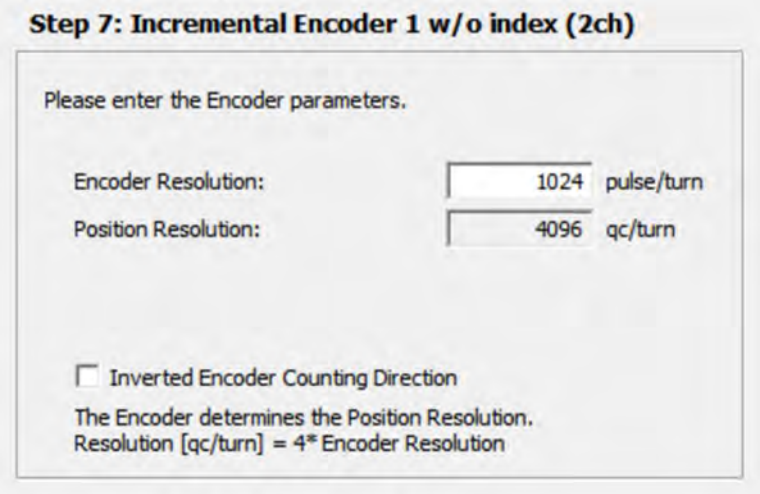

Figure 64. Gimbal motor setup (Step 7)

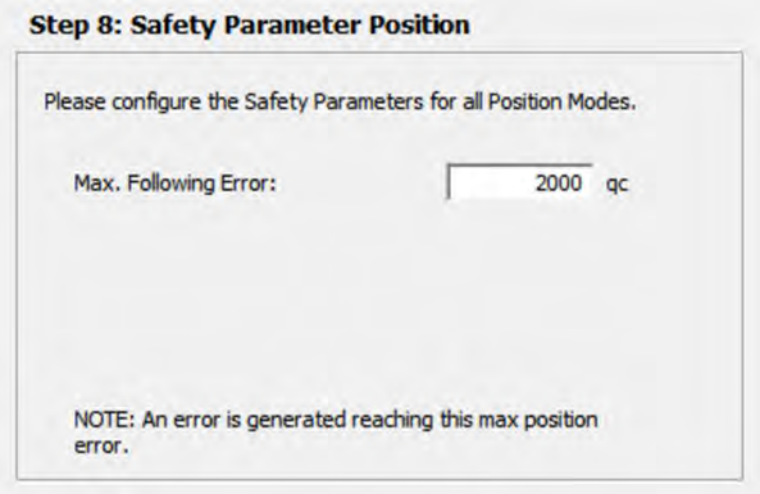

Figure 65. Gimbal motor setup (Step 8)

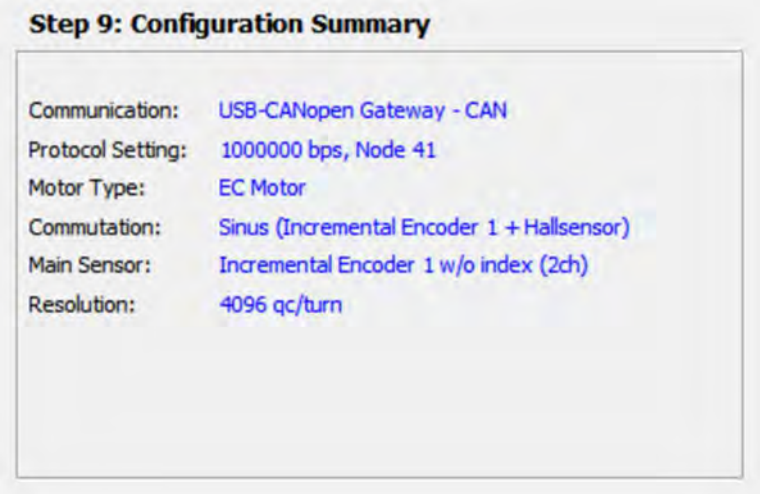

Figure 66. Gimbal motor setup (Step 9)

(2) Momentum Wheel Motor Controller Configuration

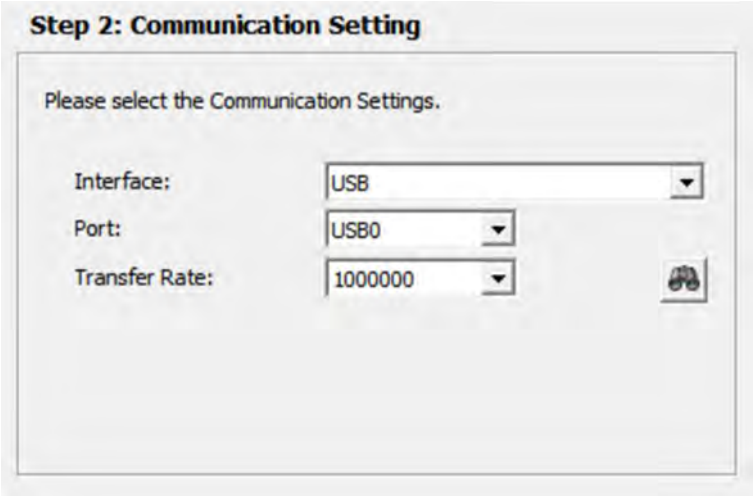

Figure 67. Momentum wheel motor setup (Step 2)

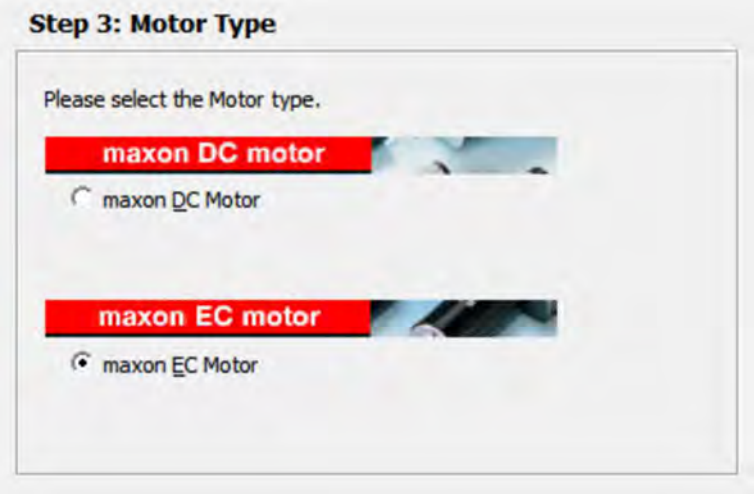

Figure 68. Momentum wheel motor setup (Step 3)

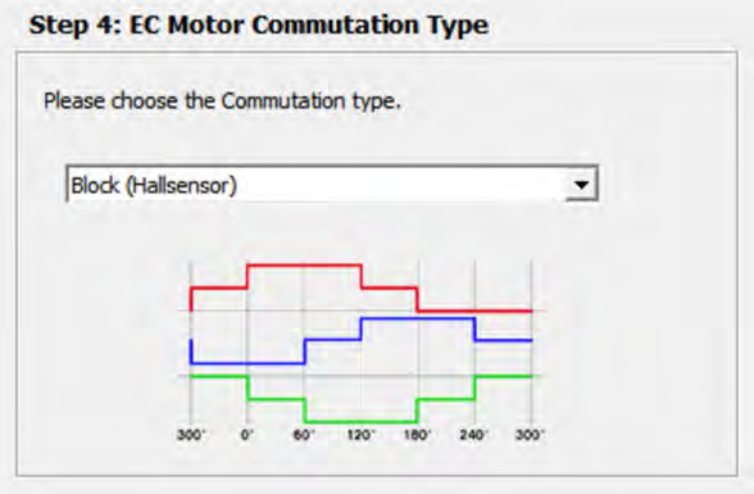

Figure 69. Momentum wheel motor setup (Step 4)

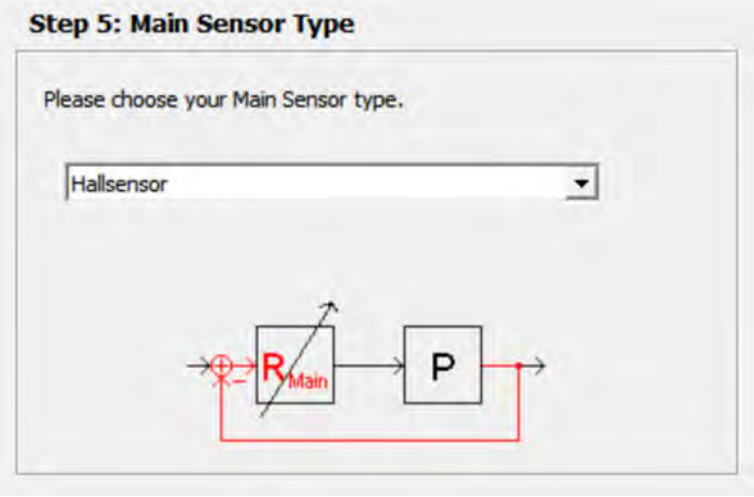

Figure 70. Momentum wheel motor setup (Step 5)

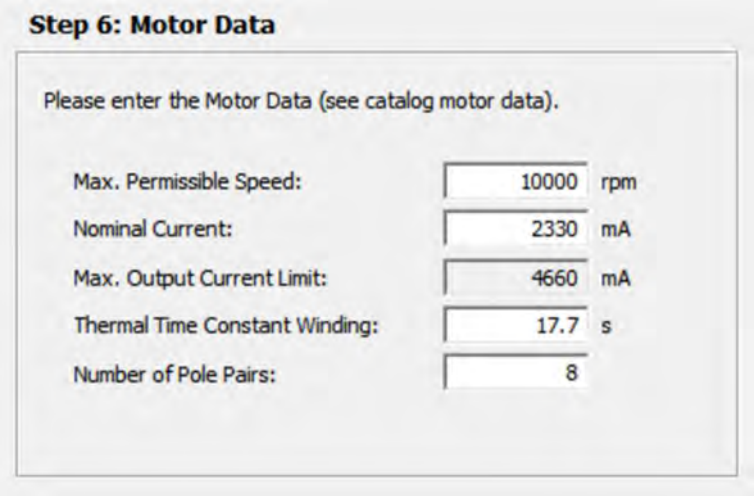

Figure 71. Momentum wheel motor setup (Step 6)

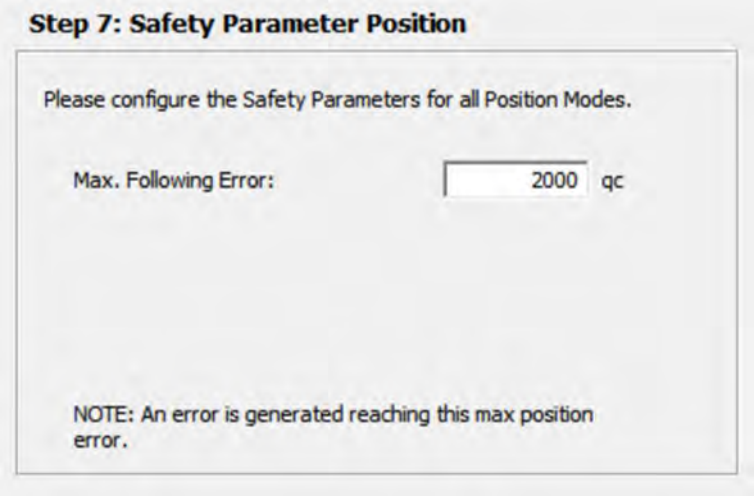

Figure 72. Momentum wheel motor setup (Step 7)

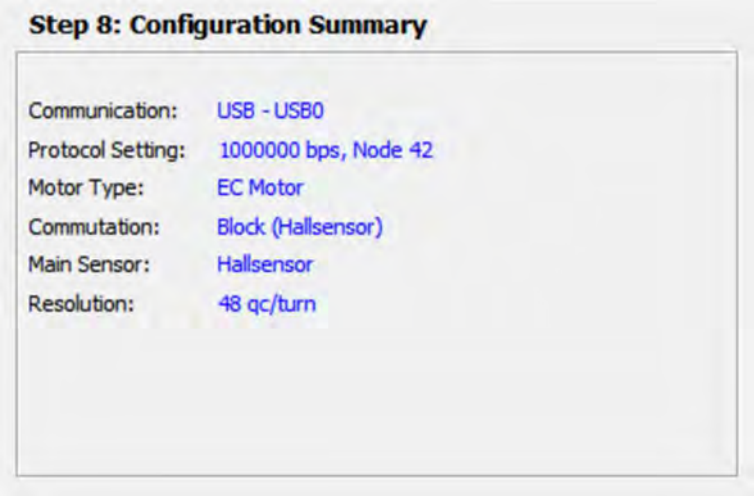

Figure 73. Momentum wheel motor setup (Step 8)

APPENDIX B. SAMPLE MATLAB CODE FOR IMPORTING AND PLOTTING REAL-TIME DATA

This appendix includes the *MATLAB* script used to import real-time data from the *LabVIEW* .LVM file. The *Reader VI.vi* is configured to save the output file as *Data.lvm* on the host workstation. This *MATLAB* script assumes that the *Data.lvm* is located in the same folder at the *MATLAB* .m file.

```
%% Data Import

Data = load('Data.lvm');

Start = 1;
Stop  = find(Data(:,2)>0,1,'Last');

Time    = Data(Start:Stop,45);
ConFreq = Data(Start:Stop,46);
CANFreq = 1./diff(Time);
q1C     = Data(Start:Stop,2);
q2C     = Data(Start:Stop,3);
q3C     = Data(Start:Stop,4);
q4C     = Data(Start:Stop,5);
q1A     = Data(Start:Stop,6);
q2A     = Data(Start:Stop,7);
q3A     = Data(Start:Stop,8);
q4A     = Data(Start:Stop,9);
wX      = Data(Start:Stop,10);
wY      = Data(Start:Stop,11);
wZ      = Data(Start:Stop,12);
uX      = Data(Start:Stop,13);
uY      = Data(Start:Stop,14);
uZ      = Data(Start:Stop,15);
d1DotC  = Data(Start:Stop,16);
d2DotC  = Data(Start:Stop,17);
d3DotC  = Data(Start:Stop,18);
d4DotC  = Data(Start:Stop,19);
d1DotA  = Data(Start:Stop,20);
d2DotA  = Data(Start:Stop,21);
d3DotA  = Data(Start:Stop,22);
d4DotA  = Data(Start:Stop,23);
d1A     = Data(Start:Stop,24);
d2A     = Data(Start:Stop,25);
```

```
d3A     = Data(Start:Stop,26);
d4A     = Data(Start:Stop,27);
d1Amps  = Data(Start:Stop,28);
d2Amps  = Data(Start:Stop,29);
d3Amps  = Data(Start:Stop,30);
d4Amps  = Data(Start:Stop,31);
w1      = Data(Start:Stop,32);
w2      = Data(Start:Stop,33);
w3      = Data(Start:Stop,34);
w4      = Data(Start:Stop,35);
w1Amps  = Data(Start:Stop,36);
w2Amps  = Data(Start:Stop,37);
w3Amps  = Data(Start:Stop,38);
w4Amps  = Data(Start:Stop,39);
HX      = Data(Start:Stop,40);
HY      = Data(Start:Stop,41);
HZ      = Data(Start:Stop,42);
qNorm   = Data(Start:Stop,43);
M       = Data(Start:Stop,44);

qC      = [q1C,q2C,q3C,q4C];
qA      = [q1A,q2A,q3A,q4A];
wBN     = [wX,wY,wZ];
u       = [uX,uY,uZ];
dDotC   = [d1DotC,d2DotC,d3DotC,d4DotC];
dDotA   = [d1DotA,d2DotA,d3DotA,d4DotA];
dA      = [d1A,d2A,d3A,d4A];
dAmps   = [d1Amps,d2Amps,d3Amps,d4Amps];
w       = [w1,w2,w3,w4];
wAmps   = [w1Amps,w2Amps,w3Amps,w4Amps];
H       = [HX,HY,HZ];

d1dt    = [0; diff(d1A)./diff(Time)];
d2dt    = [0; diff(d2A)./diff(Time)];
d3dt    = [0; diff(d3A)./diff(Time)];
d4dt    = [0; diff(d4A)./diff(Time)];

dddt    = [d1dt d2dt d3dt d4dt];

%% Data Plotting

figure(1) % Simulated Quaternion
plot(Time,qA,'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Quaternion')
legend('q_1',...
```

```
        'q_2',...
        'q_3',...
        'q_4',...
        'Location','NorthEast')
xlim([0 Time(end)])
grid on

figure(2) % Spacecraft Angular Rates
plot(Time,wBN.*(180/pi),'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Angular Velocity (Degrees/Second)')
legend('\omega_x',...
        '\omega_y',...
        '\omega_z',...
        'Location','NorthEast')
xlim([0 Time(end)])
grid on

figure(3) % Commanded Torque
plot(Time,u,'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Torque (Nm)')
legend('u_x',...
        'u_y',...
        'u_z',...
        'Location','SouthEast')
xlim([0 Time(end)])
grid on

figure(4) % Commanded vs. Sampled Gimbal Rates
subplot(1,2,1)
plot(Time,dDotC,'LineWidth',2)
set(gca,'ColorOrderIndex',1)
hold on
stairs(Time,dDotA,'-','LineWidth',1)
xlabel('Time (Seconds)')
ylabel('Gimbal Rate (Radians/Second)')
Legend = legend('$\dot{\delta}_{1, Commanded}$',...
                '$\dot{\delta}_{2, Commanded}$',...
                '$\dot{\delta}_{3, Commanded}$',...
                '$\dot{\delta}_{4, Commanded}$',...
                '$\dot{\delta}_{1, Sampled}$',...
                '$\dot{\delta}_{2, Sampled}$',...
                '$\dot{\delta}_{3, Sampled}$',...
                '$\dot{\delta}_{4, Sampled}$',...
                'Location','SouthEast');
```

```
set(Legend,'Interpreter','Latex')
xlim([Time(1) Time(end)])
xlim([Time(1) 15])
ylim([-1.1 0.4])
grid on
hold off

subplot(1,2,2)
plot(Time,dDotC,'LineWidth',2)
set(gca,'ColorOrderIndex',1)
hold on
stairs(Time,dddt,'-','LineWidth',1)
xlabel('Time (Seconds)')
ylabel('Gimbal Rate (Radians/Second)')
Legend = legend('$\dot{\delta}_{1, Commanded}$',...
                '$\dot{\delta}_{2, Commanded}$',...
                '$\dot{\delta}_{3, Commanded}$',...
                '$\dot{\delta}_{4, Commanded}$',...
                '$\dot{\delta}_{1, d\delta/dt}$',...
                '$\dot{\delta}_{2, d\delta/dt}$',...
                '$\dot{\delta}_{3, d\delta/dt}$',...
                '$\dot{\delta}_{4, d\delta/dt}$',...
                'Location','SouthEast');
set(Legend,'Interpreter','Latex')
xlim([Time(1) Time(end)])
xlim([Time(1) 15])
ylim([-1.1 0.4])
grid on
hold off

figure(5) % Gimbal Displacement
stairs(Time,dA,'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Gimbal Displacement (Radians)')
legend('\delta_1',...
       '\delta_2',...
       '\delta_3',...
       '\delta_4',...
       'Location','SouthEast')
xlim([0 Time(end)])
grid on

figure(6) % Singularity Measure
plot(Time,M,'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Singularity Measure')
```

```
xlim([0 Time(end)])
grid on

figure(7) % CAN Loop Real-Time Frequency
stairs(Time(1:end-1),CANFreq,'LineWidth',2)
xlabel('Time (Seconds)')
ylabel('Frequency (Hertz)')
xlim([0 Time(end-1)])
ylim([0 110])
grid on
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] J. R. Wertz, *Spacecraft Attitude Determination and Control*, Dordrecht, Netherlands: Kluwer Academic Publishers, 1978.

[2] V. A. Chobotov, *Spacecraft Attitude Dynamics and Control*, Malabar, FL: Krieger Publishing Company, 1991.

[3] J. R. Wertz, D. F. Everett and J. J. Puschell, *Space Mission Engineering: The New SMAD*, Hawthorne, CA: Microcosm Press, 2011.

[4] B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed., Reston, VA: American Institute of Aeronautics and Astronautics, 2008.

[5] R. Votel and D. Sinclair, "Comparison of control moment gyros and reaction wheels for small Earth-observing satellites," in *26th Ann. AIAA/USU Conf. on Small Satellites*, Logan, UT, 2012.

[6] S. R. Crews, “Increasing slew performance of reaction wheel attitude control systems,” M.S. thesis, Dept. of Mech. and Aerosp. Eng., Naval Postgraduate School, Monterey, CA, 2013.

[7] M. Macdonald and V. Badescu, *The International Handbook of Space Technology*, New York, NY: Springer, 2014.

[8] N. S. Bedrossian, S. Bhatt, W. Kang and I. M. Ross, "Zero-propellant maneuver guidance: rotating the International Space Station with computational dynamic optimization," *IEEE Control. Syst. Mag.*, vol. 29, no. 5, pp. 53-73, 2009.

[9] M. Karpenko and I. M. Ross, "Flight implementation of shortest-time maneuvers for imaging satellites*," Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1069-1079, 2014.

[10] M. Karpenko and R. J. Proulx, "Experimental implementation of Riemann-Stieltjes optimal control for agile imaging satellites," *Journal of Guidance, Control, and Dynamics*, to be published.

[11] H. Kurokawa, "A geometric study of single gimbal control moment gyros," Mechanical Engineering Laboratory, Tokyo, Japan, Tech. Rep., 1997.

[12] D. Jung and P. Tsiotras, "An experimental comparison of CMG steering control laws," in *AIAA/AAS Astrodynamics Specialist Conf. and Exhibit*, Providence, RI, 2004.

[13] B. Wie, D. Bailey and C. Heiberg, "Rapid multitarget acquisition and pointing control of agile spacecraft," *Journal of Guidance, Navigation, Control, and Dynamics*, vol. 25, no. 1, pp. 96-104, 2002.

[14] M. Karpenko and I. M. Ross, "Implementation of shortest-time maneuvers for generic CMG steering laws," in *Proc. of the AIAA/AAS Astrodynamics Specialist Conf.*, Minneapolis, MN, 2012.

[15] W. Ley, K. Wittmann and W. Hallmann, *Handbook of Space Technology*, Sussex, United Kingdom: Wiley, 2009.

[16] J. L. Schwartz, M. A. Peck and C. D. Hall, "Historical review of spacecraft simulators," in *Proc. of the AAS/AIAA Spaceflight Mechanics Meeting*, Ponce, Puerto Rico, 2003.

[17] J. J. Kim and B. N. Agrawal, "System identification and automatic mass balancing of ground-based three-axis spacecraft simulator," in *Proc. of the AIAA Guidance, Navigation, and Control Conf. and Exhibit*, Keystone, Colorado, 2006.

[18] S. Chesi, Q. Gong, V. Pellegrini, R. Cristi and M. Romano, "Automatic mass balancing of a spacecraft three-axis simulator: analysis and experimentation," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 197-206, 2014.

[19] Andrews Space, *3DOF Satellite Simulator User's Guide*, Tukwila, WA, 2009.

[20] K. L. Ackman, "Prototyping of an Open-Architecture CMG System," M.S. thesis, Dept. of Mech. and Aerosp. Eng., Naval Postgraduate School, Monterey, CA, 2012.

[21] S. M. Kocis, "Prototyping and Characterization of an Experimental Single Gimbal CMG," M.S. thesis, Dept. of Mech. and Aerosp. Eng., Naval Postgraduate School, Monterey, CA, 2015.

[22] M. J. Sidi, *Spacecraft Dynamics and Control*, Cambridge, United Kingdom: Cambridge University Press, 1997.

[23] B. Wie, H. Weiss and A. Arapostathis, "Quaternion feedback regulator for spacecraft Eigenaxis rotations," *Journal of Guidance, Navigation, Control, and Dynamics*, vol. 12, no. 3, pp. 375-380, 1989.

[24] S. J. Leon, *Linear Algebra with Applications*, 5th ed., Upper Saddle River, NJ: Prentice Hall, 2009.

[25] *Controller Area Network (CAN) Tutorial*, National Instruments, Austin, TX.

[26] *NI-CAN Hardware and Software Manual*, National Instruments, Austin, TX, 2010.

[27] *CompactRIO NI cRIO 9024 Operating Instructions and Specifications*, National Instruments, Austin, TX, 2010.

[28] *NI LabVIEW for CompactRIO Developer's Guide*, National Instruments, Austin, TX, 2014.

[29] *EPOS2 24/5 Positioning Controller Hardware Reference*, Maxon Motor AG, Sachseln, Switzerland, 2014.

[30] *EPOS2 24/5 Cable Starting Set*, Maxon Motor AG, Sachseln, Switzerland, 2013.

[31] *CompactRIO Reconfigurable Embedded Chassis Installation Instructions*, National Instruments, Austin, TX, 2009.

[32] *NI 9853 Getting Started Guide*, National Instruments, Austin, TX, 2015.

[33] *EPOS2 P 24/5 Programmable Positioning Controller Hardware Reference*, Maxon Motor AG, Sachseln, Switzerland, 2014.

[34] *EPOS2 Application Notes Collection*, Maxon Motor AG, Sachseln, Switzerland 2014.

[35] J. Travis and J. Kring, *LabVIEW for Everyone: Graphics Programming Made Easy and Fun*, Upper Saddle River, NJ: Prentice Hall, 2007.

[36] R. H. King, *Introduction to Data Acquisition with LabVIEW*, 2nd ed., New York, NY: McGraw Hill, 2013.

[37] *EPOS2 Firmware Specification*, Maxon Motor AG, Sachseln, Switzerland, 2014.

[38] *KVH DSP-3000 Fiber Optic Gyro Technical Manual*, KVH Industries, Middletown, RI, 2009.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California